Journal of Information Processing Vol.20 No.2 426–434 (Apr. 2012)

[DOI: 10.2197/ipsjjip.20.426]

Regular Paper

Efficient Shape Recognition of Dynamic Event Regions Using Wireless Sensor Networks

Satoshi Fujita[1,a) Yang Yang[1]

Received: September 29, 2011, Accepted: January 13, 2012

Abstract: In this paper, we consider the problem of recognizing the *shape of dynamic event regions* in wireless sensor networks (WSNs). A key idea of our proposed algorithm is to use the notion of *distance field* defined by the hop count from the boundary of event regions. By constructing such a field, we can easily identify several critical points in each event region (e.g., local maximum and saddle point) which could be effectively used to characterize the shape and the movement of such event regions. The communication cost required for the shape recognition of dynamic event regions significantly decreases compared with a naive centralized scheme by selectively allowing those critical points to send a certification message to the boundary of the event region and a notification message to the data aggregation points. The performance of the proposed scheme is evaluated by simulations. The simulation results indicate that: 1) the number of messages transmissions during a shape recognition significantly decreases compared with a naive centralized scheme; 2) the accuracy of shape recognition depends on the density of the underlying WSN, while it is robust against the lack of sensors in a particular region in the field, and 3) the proposed event tracking scheme correctly recognizes the movement of an event region with small number of message transmissions compared to a centralized scheme.

Keywords: wireless sensor network, shape recognition, distributed algorithm

1. Introduction

Along with recent advancements in microelectronics and wireless communication technologies, **wireless sensor networks** (WSNs) have been attracting considerable attention in the fields of network computing and distributed processing [2], [8], [14]. The primary task of a WSN is to continuously monitor the surrounding environment (e.g., the temperature and the density of NO_x in the atmosphere) to report changes in status to an appropriate data aggregation point such as a meteorological observatory, in either an event-driven or a query-based fashion.

A typical WSN is composed of a number of tiny devices called **sensor nodes** (or nodes), each of which is capable of conducting simple arithmetical operations, communicating wirelessly with nearby nodes, and sensing the status of the surrounding environment. In the *multi-hop* version of WSNs [11], [13], which is the target of the current paper, each (sensor) node plays the role of a message router in addition to the role of a sensing device. Note that in such a multi-hop WSN, all participant nodes should collaborate with each other in order to report their status to a data aggregation point in an efficient and timely manner. A number of pervasive applications automatically deploy sensor nodes providing a continuous and/or periodic snapshot of the environment, such as habitat monitoring [3], target tracking [15], aquatic observations [6], and surveillance [19].

There are several key issues in designing an available environment monitoring system (EMS) over WSNs. The first issue we need to address is how to convey a faithful representation of the signal field to an aggregation point while keeping the amount of utilized resources sufficiently low. In general, an "event" to be monitored may spread over a wide region in the given field. For example, consider the detection of gas leakage in a given field. We should first consider a WSN which is expected to report "which region in the field has a gas concentration exceeding a certain threshold." Once gas leaks out, the system should continuously monitor all points in the field to observe the spread of the gas. In a centralized approach, each node reports its status to the aggregation point as soon as it detects a change in the gas density. However, such a naive scheme does not scale, since it consumes a large amount of network resources for the communication with the aggregation point, such as CPU time, communication bandwidth, and electric power. This indicates that in order to achieve a highly scalable EMS based on WSNs, we have to develop a *distributed data aggregation scheme* such that nodes covered by an event region autonomously organize themselves and conduct an appropriate local computation to determine an extent of the region before sending a report to the aggregation point. The second issue we have to consider is concerned with the trade-off between time-criticalness and the accuracy of the monitored information. In fact, certain events in real applications such as gas leakage and fires are highly time-critical, while it is generally sufficient to know an approximated direction of the event region. On the other hand, there are other type of events which are not time-critical but require precise location information, e.g., plants growth [18] and habitat changes [10]. In such applications, we need a precise location of target events in order to reflect the acquired data to (sometimes political) decisions.

[1] Department of Information Engineering, Graduate School of Engineering, Hiroshima University, Hiroshima 724–8527, Japan

a) fujita@se.hiroshima-u.ac.jp

Motivated by those considerations, a variety of data aggregation methods for WSNs have been proposed in the literature, to acquire statistical attributes of sensed data, such as min, max, and average [7] as well as robust statistics such as median and quantiles [16]. Unfortunately however, most of those techniques have merely focused on numerical statistics and did not pay attention to the *geometric shape* of the event region, although such information is generally useful to intuitively grasp an overview of the monitored event.

In this paper, we consider the problem of recognizing the shape of an event region in WSNs. The first idea of our algorithm is to focus on a *distance field* defined by the hop count from the boundary of an event region. By constructing such a virtual field in the given space, we can easily identify several critical points in the event region (e.g., local maximum and saddle point), which well characterize the shape of the region. The communication cost required for shape recognition is significantly lower than the naive centralized scheme because it selectively allows those critical points to send a message to the aggregation point. After receiving such location information, the host behind the aggregation point can estimate the actual shape of the event region with some heuristics. The reader should note that a similar idea to identify critical points in WSNs has already been proposed in the literature [21], although it gave no experimental evaluation concerning the time and message complexity. In addition, the authors of the paper merely proposed a distributed scheme to identify critical points in static event regions and point out possible applications of the scheme including efficient message routing and facility location; i.e., they did not extend the idea to the tracking of dynamic event regions. The second idea of our algorithm is to extend the above scheme to the recognition of the move of an event region in the given space. More concretely, by flooding an inquiry message from each critical point, nodes on the boundary can certify that whether the point is still a critical one or not. Thus, if the region moves in the field, a node on the boundary can locally check the facts of the movement, and can notify it to the (former) critical point with necessary information such as the direction and the distance of the move. After receiving such notifications from the boundary nodes, a (former) critical point hands over the role of critical point to an appropriate node in the field, to achieve efficient tracking of the move of event region. The performance of the proposed scheme was experimentally evaluated by simulation. The result of the simulations is summarized as follows: 1) the number of messages transmitted during a shape recognition significantly decreases by applying the scheme compared with a naive centralized scheme; 2) the accuracy of shape recognition depends on the density of the underlying WSN, while it is robust against the lack of sensors in a particular region in the field, and 3) the proposed event tracking scheme correctly recognizes the move of event region with a small number of transmitted messages compared with a centralized scheme.

The remainder of this paper is organized as follows. Section 2 outlines related work. Section 3 describes a formal model of the WSN. Section 4 describes a shape recognition scheme for static event region, and Section 5 extends it to the recognition of a dynamic event region. The result of the simulations is shown in Section 6. Finally, Section 7 concludes the paper with topics for future work.

2. Related Work

Previous works on event shape recognition in WSN can be classified into two categories by their main techniques which are boundary detection schemes and fault-tolerant schemes.

In **boundary detection** schemes, the shape of an event region is recognized by providing a description of the boundary. Chintalapudi et al. [4] proposed a classifier-based edge detection mechanism to generate a linear polynomial representing the boundary of an event, where each node detects an edge of the region by sampling the status of its nearby nodes. Nowak et al. [12] introduced a quadtree structure to efficiently collect detected information to a central node; i.e., it recursively partitions a given space into four subspaces, and edges detected by a node corresponding to a leaf in the quadtree are collected to the root of the tree, in such a way that the shape of the boundary is recognized by the node corresponding to the root. Unfortunately however, such boundary detection schemes have a serious drawback such that nodes detecting the boundary must form a continuous loop to correctly recognize the shape of the overall region; i.e., it does not allow for missing nodes on the boundary which significantly degrades the robustness of the scheme.

Fault-tolerant schemes focus on bare analog signals received from sensors rather than a binary representation obtained through interpretations. Assuming that event measurements are spatially correlated, those schemes try to distinguish fault sensor measurements; i.e., disambiguate events by exchanging signals among nearby sensors. Krishnamachari et al. [9] proposed a scheme based on a distributed Bayesian method to detect and to correct such faults. However, such techniques cannot be applied to general WSNs since it requests each node to know its precise geographical location through expensive GPS or RF-based beacons. In addition, it requests each node to autonomously identify interesting output signals in order to recognize the shape of event region merely through a collaboration among sensors.

3. Model

This section describes a model of WSN which will be used throughout of this paper. Let V be a set of sensor nodes. Each node in V is capable of sensing the environment via attached sensor devices, communicating with nearby nodes via wireless communication device, and conducting simple computations with a tiny CPU and a small memory. In the following, we assume that each node in V has a unique ID, is located on a two-dimensional plane and is associated with a point in a two-dimensional coordinate space. Let $p(u)$ denote the point associated with node u. Note that $p(u)$ is a variable used only in the explanation of algorithms and we do not allow each node u to refer to its precise location $p(u)$.

For each $u \in V$, let $N(u)$ denote the set of neighbors of u which is defined by the Euclidean distance of the corresponding points; i.e., for any $u, v \in V$, $v \neq u$, $v \in N(u)$ iff $|p(u) - p(v)| \leq 1$, where $|p - q|$ represents the Euclidean distance between points

p and q [*1]. In the proposed scheme, we assume that each node u knows a set of neighbors $N(u)$, and that a message transmitted by u is always received by all nodes in $N(u)$ in a single step, where each message has a fixed length of $O(\log |V|)$. In addition, u detects an *event* occurred in the environment if the event region covers point $p(u)$, where the **event region** is a finite region in the two-dimensional space concerned with the event (note that term "finite" means that it has a finite boundary).

Let $E \overset{\text{def}}{=} \{(u, v) \in V \times V : v \in N(u)\}$ be a symmetric relation defined by the sets of neighbors. A pair of V and E naturally defines an undirected graph G, where in the following we assume that G is connected, without loss of generality. Let σ_1, σ_2, and σ_3 be three external nodes called sinks (note that $\sigma_i \notin V$ for each i). Those sinks are used for data aggregation and node localization (detailed procedure for such operations will be described later). Throughout this paper, we assume that each sink σ_i has at least one neighbor contained in V, and knows its precise location $p(\sigma_i)$. In addition, sink σ_1 is connected to a host via a wired link. Users can interact with the WSN through the host or in other words can can send queries to the host and they can receive necessary information through the host. The main task of the host is to conduct shape recognition (i.e., an approximation of the shape of an event region) based on the information received from nodes in the WSN. The other operations are executed by individual sensor nodes in the WSN, in a distributed and autonomous manner.

4. Shape Recognition of Static Event Region

This section describes a scheme for recognizing the shape of a static event region, which will be used as a basic procedure in an *event tracking* scheme proposed in the next section. The scheme consists of the following five parts: 1) preprocessing (Section 4.1), 2) event detection (Section 4.2), 3) distance field construction (Section 4.3), 4) identification of critical points (Section 4.4), and 5) event region approximation (Section 4.5).

4.1 Preprocessing

At first, each sink broadcasts a message to initiate the calculation of the minimum hop count from the sink to each node. We can use the Dijkstra's method [1] as a concrete procedure for such calculation. After completing the calculation, each node u in the field stores tuple $\langle d_1(u), d_2(u), d_3(u)\rangle$ to its local memory, where $d_i(u)$ denotes the minimum hop count from u to sink σ_i. In the proposed scheme, such a tuple is used to approximate the location of each node in the coordinate space by assuming that the Euclidean distance is well approximated by the hop count (we will discuss this issue in Section 4.5.2). In addition, the i^{th} element of the tuple is used to navigate message transmissions towards sink σ_i through a shortest path; i.e., each node may simply forward a message towards a descent direction of the i^{th} element in the tuples.

4.2 Event Detection

Suppose that node u detects an event with event region R. After letting $r(u) :=$ true, node u notifies the fact to sink σ_1 through a shortest path as described above, where $r(u)$ is a local variable representing whether u is covered by an event region. Upon receiving a notification, sink σ_1 sends a Reply message along the above forwarding path in the reversed direction. Let u^* be the final receiver of the Reply message. Note that $r(u^*) =$ true must hold valid since it is the originator of a notification message. In the proposed scheme, even if σ_1 receives several notification messages from different originators, it sends exactly one Reply message to those notifications assuming that R consists of a single connected region (a generalization to the case of several connected regions is left as a future work).

In general, several nodes covered by a region may try to send a notification at almost the same time. In practice, such redundant message transmissions could be reduced by using the following rules: 1) two notifications are merged if they encounter each other; and 2) a message is suppressed at a node if the node has already forwarded another notification. The reader should note that such merger of notifications does not increase the length of messages since each intermediate node may simply notify its ID to a node on the shortest path to the sink to receive a Reply message.

4.3 Distance Field Construction

Let $h(u)$ denote the **height** of node u with respect to event region R, which is defined as follows:

$$h(u) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } r(u) = \textsf{false}, \text{ and} \\ \min_{v \in N(u)}\{h(v)\} + 1 & \text{otherwise.} \end{cases}$$

$h(u)$ represents the minimum hop count from the boundary of R to node u (we say that node u is at the boundary if $h(u) = 1$). A concrete procedure to calculate the height of nodes is described as follows:

- **Initialization:** If $r(u) =$ true then node u initializes $h(u)$ to one, and otherwise initializes it to zero. (Each node conducts this operation when it detects an event.)
- **Trigger:** Let u^* be the final receiver of a Reply message issued by sink σ_1. After receiving a Reply message, u^* broadcasts Inside message to all nodes in $N(u^*)$ to start a calculation of the height of nodes.
- **Propagate:** If node v receives Inside message from its neighbor for the first time, and if $r(v) =$ true, then it broadcasts Inside to $N(v)$.
- **Update:** If node v receives Inside message from all nodes in $N(v)$ and if $h(v) = 1$, then it updates $h(v)$ to two, and broadcasts Height(2) message to $N(v)$. Similarly, for each $i \geq 2$, if v receives Height(i) message from all nodes in $N(v)$ and if $h(v) = i$, then it updates its height to $i + 1$, and broadcasts Height($i + 1$) message to $N(v)$.

4.4 Identification of Critical Points

The basic idea of our scheme is to recognize the shape of an event region through identifying a collection of critical points in the distance field. More concretely, we adopt local maxima and saddle points as the critical points characterizing the shape of the event region.

[*1] We assume that the transmission radius of wireless communication device is a unit distance, for simplicity.

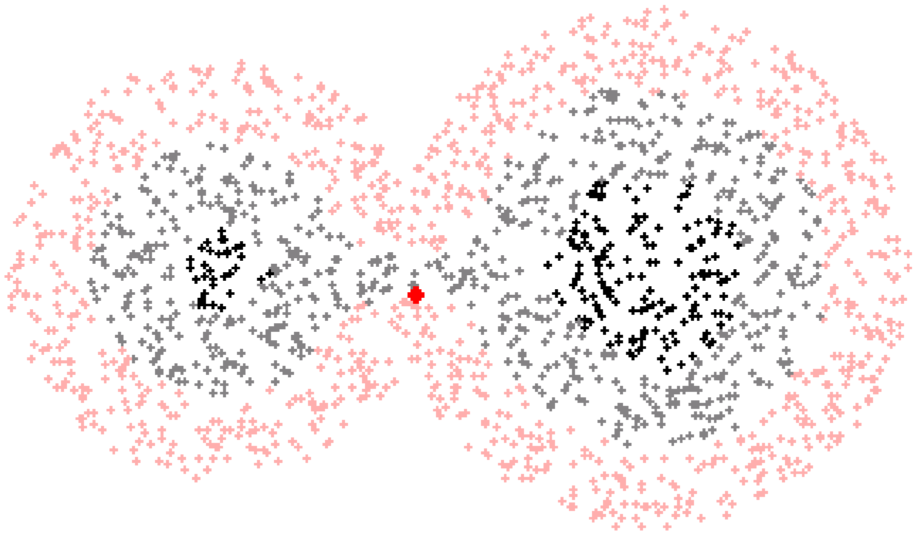

Fig. 1 Critical points in an event region. A saddle point is represented as a big red dot, and nodes with larger h-values than the saddle point are denoted by dark colors.

A node can easily recognize itself as a local maximum if the height of the node is h and it does not receive a Height($h + 1$) message from its neighbor (or if the height of the node is one and it does not receive an Inside message from its neighbor) for a certain period of time. In contrast, identifying saddle points is not as easy as identifying local maxima since since a saddle point is simultaneously adjacent with a node with higher h-value and a node with lower h-value; i.e., each node cannot decide whether it is a saddle point or not, by merely observing h-values in its neighborhood. **Figure 1** illustrates a saddle point in an event region where the large red dot represents a saddle point.

In order to overcome such difficulty in calculating saddle points, we apply a sweep method proposed by Skraba et al. [17] Let U ($\subseteq V$) be a set of identified local maxima, and $G(U)$ be a subgraph of G induced by U. For each connected component in $G(U)$, we select an arbitrary node in the component as an initiator of the sweep process (note that we can use a unique ID given to each node for such an arbitration). Let U' be the set of initiators. At first, each initiator $u \in U'$ broadcasts a Sweep message containing the name and the height of node u, to all neighbors in $N(u)$. This message is propagated to all nodes covered by the same event region R, by forwarding the message towards a descending direction of the h-value. Note that each node that received a Sweep message knows the name and the height of the corresponding local maximum. If v receives at least two sweep messages originating from different local maxima, v recognizes itself as a *candidate* of saddle point, and executes the following operations in order to certify whether it is an *actual* saddle point: 1) stop the forwarding of received Sweep messages to the next node; and 2) broadcast a message containing $h(v)$ to nodes in $N(v)$. Any candidate saddle point v that does not receive a broadcast message containing an h-value not smaller than $h(v)$ from its neighbor, it can recognize itself as an actual saddle point.

Although the above scheme certainly identifies several critical points in a fully distributed manner, it often (mis)identifies several (non-critical) nodes as local maxima if the density of nodes is low. More specifically, it does not guarantee that node v always has a neighbor u such that $h(u) = h(v) + 1$ when v is not local maximum; i.e., v may (mis)identify itself as a local maximum when each neighbor w of v has the same (or lower) height with v, and when it has the same height with v, it is adjacent with a node with height $h(v) + 1$. Such a (mis)identification could be partially resolved in the following manner (in the following, we call it a **pruning process**): For each node u, if it identifies itself as a local maxima, it transmits a LocalMax message including $h(u)$ to $N(u)$ (more precisely, a LocalMax message should be divided into several packets since we are assuming that the length of each message has a fixed length). Upon receiving a LocalMax message, node v transmits an NG message to u, if there are any $w \in N(v)$ such that $h(w) > h(u)$. If node u receives an NG message, it excludes itself from a set of local maxima.

4.5 Event Region Approximation

After recognizing itself as a critical point, node u transmits an Event message towards sink σ_1 in order to notify the following information to the host: 1) tuple $\langle d_1(u), d_2(u), d_3(u)\rangle$ indicating an approximated location of node u; 2) the height $h(u)$ of node u, and 3) if u is a saddle point, it designates local maxima corresponding to the saddle point.

4.5.1 Outline

After collecting Event messages received from critical points through sink σ_1, the host conducts a shape approximation in the following three steps:

- **Step 1:** At first, the host *imaginarily* associates each critical point u to a point $\tilde{p}(u)$ in the two-dimensional coordinate space. Recall that since we are assuming that no node in V knows its precise location, the host must *estimate* the location of each node from a tuple of hop counts to three sinks from the node (a concrete procedure for the estimation will be described later).
- **Step 2:** For each local maximum u, the host utilizes a phantom circle centered at $\tilde{p}(u)$ with radius $(h(u) - 1) \times \alpha$, where α (≤ 1) is an expected distance covered by each communication hop (see Section 4.5.2 for the details), and regards the

area spanned by such circles as an approximation of event region R.

- **Step 3:** Finally, the host conducts an adjustment of the resultant shape by using saddle points in the following manner: 1) let w be a saddle point corresponding to local maxima u and v; 2) it considers a circle of radius $(h(w) - 1) \times \alpha$ centered at point $\tilde{p}(w)$; and 3) includes the minimum convex region containing circles corresponding to w and u to the recognized shape, and conducts the same operation for circles corresponding to w and v.

Example: Suppose that the host receives the following three Event messages from critical points u, v, and w:

$$(u, \langle 6, 5, 11\rangle, 4, -), \quad (v, \langle 12, 10, 6\rangle, 5, -),$$
$$\text{and } (w, \langle 10, 8, 9\rangle, 3, \langle u, v\rangle).$$

The first field in the message represents the originator of the message, and the second field represents the approximated location of the originator. The third field represents the height of the originator, and the fourth field designates a list of corresponding local maxima, which is empty if the originator is a local maximum.

After receiving those messages, the host conducts the following operations. In Step 1, it associates three nodes u, v, and w to points in the two-dimensional space. Second, it calculates a union of circles centered at local maxima or in other words calculates a circle of radius $(h(u) - 1) \times \alpha = 3\alpha$ centered at point $\tilde{p}(u)$, a circle of radius $(h(v) - 1) \times \alpha = 4\alpha$ centered at point $\tilde{p}(v)$, and regards a union of them as an approximation of the event region. Finally, it adjusts the resultant shape by using saddle point w.

4.5.2 Location Estimation

Let $p(\sigma_i)$ be the coordinate point associated to sink σ_i, and let $\tilde{p}(u)$ be a variable representing an approximated coordinate point of node $u \in V$. In our model of WSN, two nodes can directly communicate with each other if and only if the Euclidean distance between them is smaller than or equal to one. Thus, $d_i(u) \geq |p(u) - p(\sigma_i)|$, and the difference between $d_i(u)$ and the Euclidean distance increases as the density of the nodes in the field diminishes. In order to tolerate such an inaccuracy, we propose following operation in the location estimation step:

- At first, we estimate the Euclidean distance between u and σ_i as $d_i(u) \times \alpha$, where α is an *expected* Euclidean distance covered by a single hop which is estimated by sampling the Euclidean distance between sinks; i.e., it is calculated as:

  $$\alpha = \frac{1}{6} \sum_{i \neq j} \frac{|p(\sigma_i) - p(\sigma_j)|}{d_i(\sigma_j)}$$

  Recall that the value of α is also used to estimate the radius of circles centered at critical points.
- Let $p_{i,j}(u)$ and $p'_{i,j}(u)$ be two points at distance $d_i(u) \times \alpha$ from σ_i and at distance $d_j(u) \times \alpha$ from σ_j (note that $p_{i,j}(u) = p'_{i,j}(u)$ if and only if the Euclidean distance between those sinks is equal to $(d_i(u) + d_j(u)) \times \alpha$). Let $Q = \{p_{i,j}(u), p'_{i,j}(u) \mid 1 \leq i < j \leq 3\}$. In order to calculate a point corresponding to node u, the host first conducts a clustering of points contained in Q using a nearest neighbor method with an appropriate threshold. If we have a cluster of size three or more, we regard the centroid of those points as a candidate for $\tilde{p}(u)$, where if we have two such clusters, we cannot determine which is an appropriate candidate in our model; i.e., it outputs two points as the candidate for $\tilde{p}(u)$ (in the following experiments, for simplicity we assume that it can always determine a unique candidate as a result of the clustering).

5. Event Tracking Scheme

5.1 Overview

In this section, we propose a distributed scheme to track the movement of an event region using the shape recognition scheme described in the last section. In this section, we assume that the shape of a given region is convex, and it does not change during the computation. In the proposed scheme, an identified local maximum periodically verifies its maximality, and if it detects that it is no longer a local maximum, it hands over the role of local maximum to an appropriate node in the field. More specifically, it periodically transmits a short message towards nodes at the boundary of the event region, and if a node receiving the message detects the change of the boundary, it notifies that fact to the (former) local maximum.

Key points of such a verification-based scheme are as follows: 1) how to disseminate a verification message towards the boundary of a region and how to collect notification from them; 2) how to detect the change of the boundary merely by referring to the local information; and 3) how to estimate the direction and the distance to the new local maximum from the collected notification messages. In the following explanation, we use symbol R to denote the region before a change, and R' to denote the region after the change. Recall that $h(v)$ denotes the height of node v in R. In the following, a new type of message Height(i, j) is introduced to notify that the height of a node is i in R and is changed to j in R', where j takes one of the following three values: 1) $j = 0$ if the node is no longer covered by R'; 2) $j = 1$ if it is a boundary node in R'; and 3) $j = \infty$ if it is inside of R' but is not a boundary node.

5.2 Update of the Boundary

Recall that U denotes a set of local maxima which transmit Event messages to sink σ_1, and U' ($\subseteq U$) denotes a set of nodes who initiated a sweep process to identify saddle points. In the following, for simplicity, we assume $U = U'$ or in other words that U is an independent set of graph G.

Propagation: After transmitting an Event message, each $u \in U'$ waits for a certain time τ, and then starts verifying the event region R. More concretely, it transmits an Update message to $N(u)$, which is propagated to nodes covered by R, in a descent direction of h-values, or namely in a similar way to the sweep message while it is not blocked at saddle points. If it receives an Update message from a neighbor with a positive h-value for the first time, node v transmits a copy of the message to $N(v)$ with Boolean variable $r'(v)$ which indicates whether v is covered by the new event region R'. Note that by collecting all Update messages received from neighbors, v can identify itself as a boundary node or not; i.e., if it receives an Update message with $r'(w) =$ false from neighbor w and if $r'(v) =$ true, then it identifies itself as a boundary node in R'.

Note that during the propagation of Update messages, each node v knows a neighboring node with higher h-value than v in R. In the proposed scheme, we regard such a node as a **parent** of v in the distance field, where a saddle point has several parent nodes.

Collection: After receiving Update messages from all neighbors, node v at either the boundary of R or the boundary of an intersection of R and R' transmits a Height message to $N(v)$ which will be forwarded to the initiator of the Update message along a tree used for the propagation in the reverse direction (note that each node received an Update message has known its parent in the delivery tree). More concretely, 1) node v transmits message Height$(h(v), 0)$ if $h(v) \geq 1$ and it becomes an outside node of R', 2) transmits Height$(h(v), 1)$ if $h(v) \geq 1$ and it becomes a boundary node in R', and 3) transmits Height$(1, \infty)$ if $h(v) = 1$ and it becomes an interior node in R' besides the boundary.

In order to reduce the communication cost, during such transmissions several redundant messages are discarded at intermediate nodes, according to the following rules: 1) for any i and j, Height(i, j) is transmitted at most once, i.e., latter ones are simply discarded; 2) if a node receives both Height$(i, 0)$ and Height$(i', 1)$, then the former one is discarded; and 3) if it receives both Height$(i, 1)$ and Height$(i', 1)$, $i < i'$, then the former one is discarded.

5.3 Event Tracking

After receiving Height message from all neighbors, local maximum u of event region R conducts the following operations:

Case 1: If it receives Height$(1, 1)$ from all nodes in $N(u)$, u recognizes $R' = R$, and notifies the fact to the host. It then starts the next verification process after waiting for τ time.

Case 2: If it receives Height$(i, 1)$ and Height$(1, \infty)$ from different nodes in $N(u)$, u recognizes that the local maximum corresponding to u moves to a node u' in event region R', and that node u' exists in the direction of a node who sent Height$(1, \infty)$ to u. Thus, it tries to hand over the role of local maximum to u' according to the procedure described below.

Case 3: Otherwise, node u gives up to find its successor in R, and asks the host to start a new shape recognition process.

Detailed procedure for Case 2 is described as follows. Let i^* be the maximum value of i contained in messages Height$(i, 1)$ which are received by u from its neighbors. In the proposed event tracking scheme, we use $i^* - 1$ as the hop count from u to u', where u' is the successor of u in R'. The direction of u' from u is determined by constructing a distance field from terminals of an "arc of nodes" that transmitted a Height$(1, \infty)$ message towards node u. More concretely, a node who transmitted Height$(1, \infty)$ is regarded as a terminal of the arc, if it receives a Height$(1, 1)$ message from its neighbor in the collection phase. If there are several nodes satisfying the above condition in its neighbor, the node with a largest ID is selected as the terminal node. Since we are assuming that R is convex, there exist at most two terminals on the arc (if it can identify no terminal, node u gives up trying to find its successor, and asks the host to start the next shape recognition process as in Case 3). Let w_1 and w_2 be two terminals of the arc. After transmitting their Height$(1, \infty)$ messages, those nodes initiate construction of a distance field by transmitting a short message, which will be forwarded by nodes who have already forwarded a Height$(1, \infty)$ message, towards node u.

The direction of successor u' from u is identified by nodes which have the same hop count to two terminal nodes. Thus, by forwarding a hand-over message along a path consisting of such nodes for $i^* - 1$ hops, node u can successfully send a hand-over message to a candidate node of local maximum in the new event region R'. After receiving a hand-over message, node u' constructs a distance field originating from it, and starts a verification of the maximality in a similar way to the above procedure.

6. Simulation

6.1 Setup

We evaluate the performance of the proposed scheme by simulation. In the simulation, we assume that any message transmitted by node u is correctly received by all nodes in $N(u)$. To also simplify the exposition, we assume that the host can estimate the correct location of each critical point from the tuple received from it (several figures estimated by the host will be shown below). In fact, the result of the following experiments indicates that the difference between estimated and actual positions of identified critical nodes is smaller than the difference between calculated and actual critical nodes in the given event region.

A square region of size 20×20 is given as the field of events where a unit distance is defined to be the transmission radius of each node. Coordinate points of four corners of the region are $(0, 0)$, $(0, 20)$, $(20, 20)$, and $(20, 0)$ in a clockwise direction starting from the left bottom. We then select n random points in the field, and associate them to individual sensor nodes (note that each point is represented as a pair of reals and we assume that any two points are distinct, without loss of generality). Parameter n is appropriately determined in the simulation in such a way that the resultant graph is connected (i.e., an n that is too small disconnects the underlying sensor network).

As the concrete event region, in the following experiments, we consider the following two event regions R_1 and R_2, where: 1) R_1 is a simple circle with radius 4.2 centered at point $p_1 = (12, 11)$; and 2) R_2 is the face of the mouse located at the center of the square region.

6.2 Impact of Network Density

We first evaluate the impact of the density of nodes. In the following figures, the boundary of an event region is represented by a thick black line and each node in the field is represented by a gray circle of radius 0.5 (see **Fig. 2** for illustration). Thus, the reader can easily check the connectivity of the resultant graph G since two nodes are connected by an edge in G if the corresponding circles have a non-empty intersection. Yellow regions shown in Fig. 2 are outputs of the proposed scheme for event regions R_1 ((a) and (b)) and R_2 ((c) and (d)). Red dots in the figure indicate local maxima calculated by the scheme, and black dots indicate saddle points each of which is identified by two sets of connected local maxima through sweep process. Figures 2 (a) and (c) are outputs of sparse WSN consisting of 900 nodes, and Figs. 2 (b) and (d) are outputs of dense WSN consisting of 1,800 nodes. In

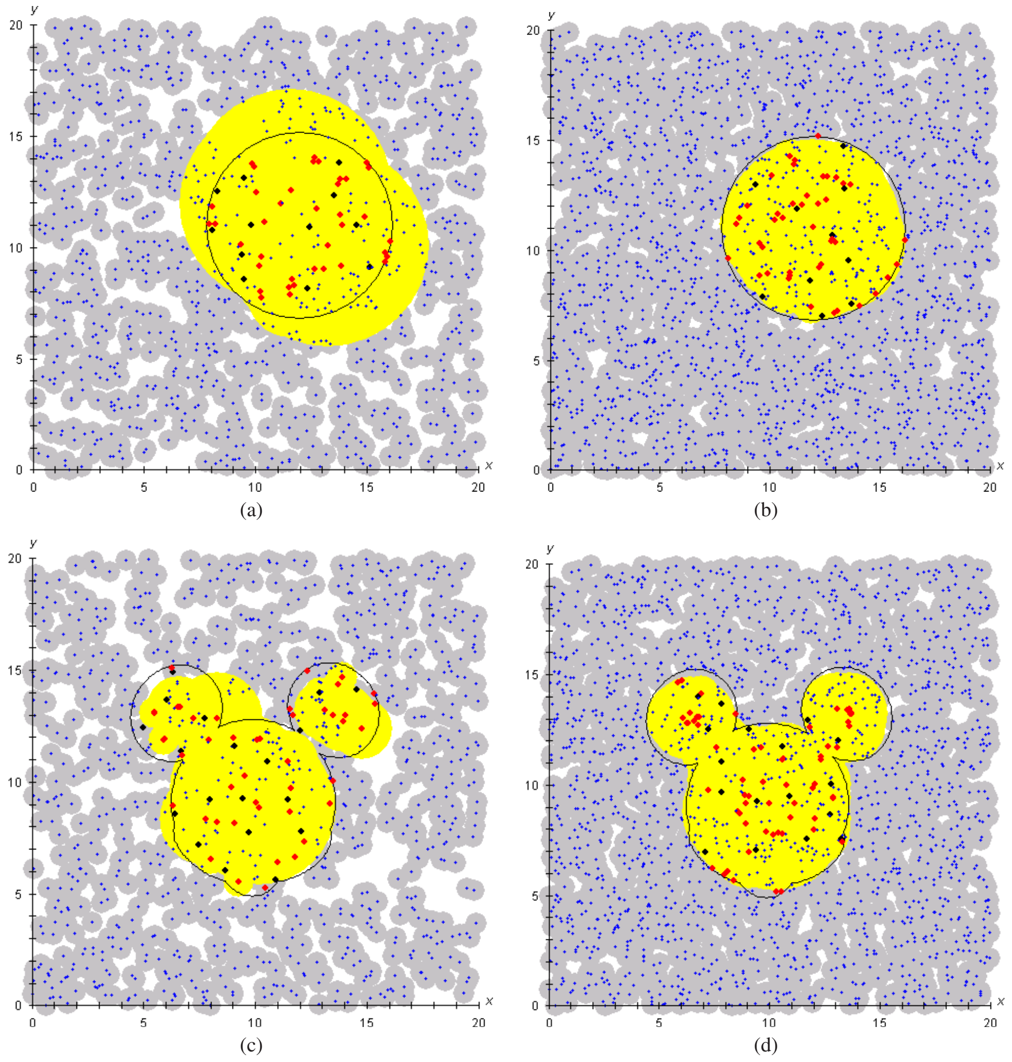

Fig. 2 The shape of event region R_1 estimated by the proposed scheme with WSN consisting of (a) 900 nodes and (b) 1,800 nodes; and R_2 estimated with WSN consisting of (c) 900 nodes and (d) 1,800 nodes.

the former case, each node has six neighbors on average, and in the latter case each node has 13 neighbors on average. It is immediate from the figures that the accuracy of the approximation increases along with the increase in density of the underlying WSN.

The inaccuracy of the approximation for the sparse case is apparently due to the lack of connectivity in the underlying graph G. In fact, if the number of nodes is small, then the nodes in the field cause a number of "holes" even if the overall graph G is kept to be connected. The size and the number of such holes increase as the number of nodes decreases. Due to the such holes, 1) several nodes calculate their height values through a long path around the hole, and 2) several nodes are (mis)identified to be local maxima due to the lack of nodes in their neighborhood. In our scheme, the first effect of holes is the primary reason for the inaccurate approximation. In Fig. 2 (a), the approximation deviating outside of the boundary of R_1, and in Fig. 2 (c), the over approximation of the ears of mouse, are all due to such a reason. On the other hand, the scheme actually (mis)identifies a number of nodes as local maxima in both of the sparse and the dense cases in Fig. 2. However, it is worth nothing here that even if it could unexpectedly identify nodes as critical points, the shape estimated by the scheme is not significantly affected, since it takes a union of circles centered at identified points (any node outside the region does not identify itself as a boundary node).

6.3 Efficiency of Static Shape Recognition

Next, we evaluate the efficiency of our shape recognition scheme in terms of the number of transmitted messages. Each data described below is an average taken over 10 experiments (each instance is randomly generated by selecting n random points in the given field, while the location of sinks is fixed). Recall that in our scheme, a shape recognition is initiated by a node detecting an event by transmitting a notification which will be forwarded to sink σ_1, and terminates when σ_1 receives all Event messages concerned with the event. We count the transmission of messages with respect to the notifications, Reply messages from σ_1, Inside messages, Height messages, Sweep messages and Event messages, and for the third scheme, we also count the number of LocalMax and NG messages. In the following, we assume that sink σ_1 is placed at point $(0, 20)$.

Figure 3 compares the result on three schemes. The first one is

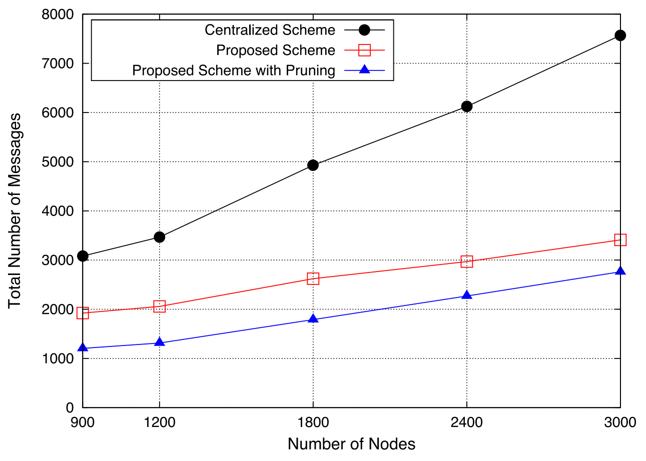

Fig. 3 A comparison of the total number of messages transmitted during shape recognition procedure of centralized scheme, proposed scheme and the improvement.

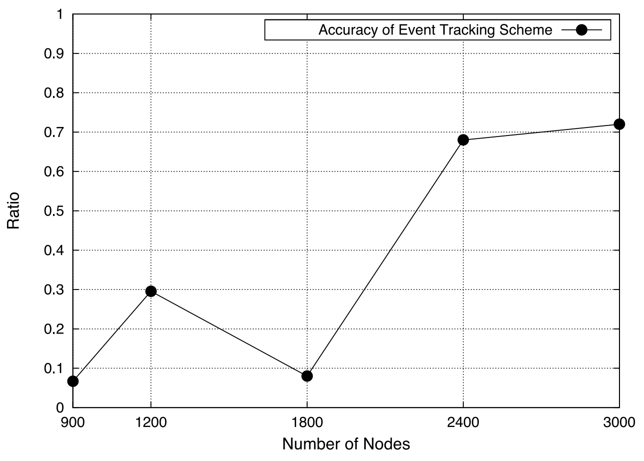

Fig. 4 Accuracy of our event tracking scheme.

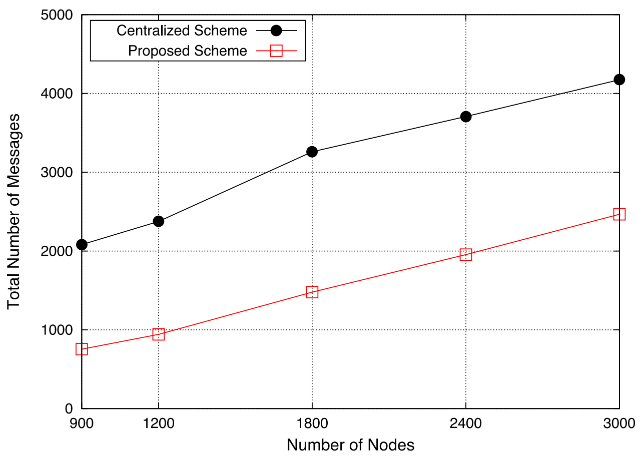

Fig. 5 A comparison of the total number of messages transmitted during the event tracking procedure of proposed scheme and centralized scheme.

a centralized scheme, the second one is our original scheme, and the third one is our improved (i.e., pruning) scheme. As expected, the second scheme certainly reduces the number of messages of the first scheme, and the third scheme further improves the second scheme. A detailed analysis of the simulation result indicates that the number of several messages does not change with such improvement; for example, in both of the second and the third schemes, as increasing n from 900 to 3,000, 1) the number of notifications similarly increases from 203.7 to 842.4; 2) the number of Reply messages takes a constant value 22.6; 3) the number of Inside and Sweep messages increases from 127.7 to 416.6; and 4) the number of Height messages similarly increases from 193.8 to 472.3. However, it does definitely reduce the number of Event messages transmitted in the second scheme; namely it reduces from 1,246.7 to 443.6 when $n = 900$, and it reduces from 1,244.5 to 173.6 when $n = 3{,}000$. Although LocalMax and NG causes an additional cost such as 85.5 for $n = 900$ and 423.4 for $n = 3{,}000$ (recall that those messages are newly introduced to the third scheme, and are proportional to the number of neighbors of each node), a significant reduction of Event makes the total cost of the third scheme lower than the second scheme for every n.

6.4 Event Tracking Scheme

In this subsection, we evaluate the performance of the proposed event tracking scheme. We consider a scenario in which event region is initially given as R_1, and after completing a shape recognition of R_1, the center of the region "moves" from $(12, 11)$ to $(10, 10)$. In the following, we refer to the region after the move as R'_1, and represent the height of node v in region R'_1 as $h'(v)$.

Let u be a local maximum of R_1 identified by our shape recognition scheme, and u^* be the successor of u calculated by our event tracking scheme (note that the scheme might not find such u^* if the density of the network is low). Now let us define the **accuracy** of selecting u^* as a successor of u as: 1) $h'(u^*)/\max_v\{h'(v)\}$ if such u^* is identified by the scheme, and 2) 0 otherwise. **Figure 4** shows how the accuracy of our proposed scheme varies by increasing the number of nodes in the network where each value is an average over 10 random instances as before. Although it takes a small value for small n, which is mainly due to failure to identify the successor u^* (recall that the accuracy takes value zero for such case), the accuracy gradually approaches to 0.72 for sufficiently large n; i.e., it correctly identifies the direction of the move and at the same time, the distance to the new local maximum is estimated almost correctly. The reason why the accuracy for 1,800 nodes is worse than the accuracy for 1,200 nodes is that the scheme could not identify the next node in many instances for 1,800 nodes (recall that the accuracy takes value 0 if it could not identify the next node).

Finally, we evaluate the efficiency of the proposed event tracking scheme in terms of the number of all transmitted messages. Recall that in our scheme, an event tracking is initiated by each local maximum u by transmitting an Update message to its neighbors, and terminates when the verification of maximality of successor u' completes. As a competitor, we consider a naive scheme which conducts the following steps in a centralized manner: 1) sink σ_1 collects location information from "all" nodes who detect the change of the coverage by the event region (i.e., node v can recognize itself as a member of $R_1 - R'_1$ if it observes a change of $r(v)$ from true to false, and a member of $R'_1 - R_1$ if it observes a change of $r(v)$ from false to true), and 2) the host estimates the current shape by those differential information and the previous shape R_1. **Figure 5** shows the result. As shown in the figure, the proposed scheme significantly improves the performance of the naive centralized scheme; e.g., it reduces the number of messages to 36% for $n = 900$, and to 60% for $n = 3{,}000$, while the cost of the boundary updating procedure is almost the same as the verification of the maximality of u', which takes 295.3 for $n = 900$ and 1,017.9 for $n = 3{,}000$.

7. Concluding Remarks

In this paper, we propose a distributed scheme to recognize the shape of a dynamic event region by WSNs. The proposed scheme significantly reduces the number of transmitted messages by identifying critical points in the given event region, and by repeatedly verifying the criticalness of such identified points. The result of the simulations indicate that the proposed scheme (almost always) correctly identifies the direction and the distance of a movement of event region, and the number of message transmissions is sufficiently small compared with a centralized scheme which collects differential information from the nodes.

A future problem is to extend the scheme such that: 1) the shape of the target event region is not restricted to be convex; 2) the efficiency of the scheme is further improved; and 3) the shape of the region can change during a recognition process (the current version allows the movement of an event region, but does not allow a change in the shape). We are planning to implement the proposed scheme in actual WSNs consisting of hundreds of sensor nodes, and apply it to actual applications.

Reference

[1] Bertsekas, D. and Gallager, R.: *Introduction to Algorithms*, Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (Eds.), pp.595–601, MIT Press and McGraw-Hill (2001).

[2] Bonnet, P., Gehrke, J.E. and Seshadri, P.: Querying the Physical World, *IEEE Personal Communications*, Vol.7, No.5, pp.10–15 (2000).

[3] Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M. and Zhao, J.: Habitat Monitoring: Application Driver for Wireless Communications Technology, *Proc. 2001 ACM SIGCOMM Workshop Data Communication in Latin America and the Caribbean*, pp.20–41 (2001).

[4] Chintalapudi, K.K. and Govindan, R.: Localized Edge Detection in Sensor Fields, *Proc. 1st IEEE International Workshop on Sensor Network Protocols and Applications* (*SNPA*), pp.59–70 (2003).

[5] Dey, T.K., Giesen, J. and Goswami, S.: Shape Segmentation and Matching with Flow Discretization, *Algorithms and Data Structures*, LNCS 2748, pp.25–36 (2003).

[6] Dhariwal, A., Zhang, B., Stauffer, B. and Oberg, C.: NAMOS: Networked Aquatic Microbial Observing System, *Proc. 2006 International Conference on Robotics and Automation* (*ICRA 06*), pp.4285–4287 (2006).

[7] Greenwald, M. and Khanna, S.: Power-Conserving Computation of Order-Statistics over Sensor Networks, *Proc. 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (*PODS*), pp.275–285 (2004).

[8] Intanagonwiwat, C., Govindan, R. and Estrin, D.: Directed Diffusion: A scalable and robust communication paradigm for sensor networks, *Proc. 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking* (*MobiCom*), pp.56–67 (2000).

[9] Krishnamachari, B. and Iyengar, S.: Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks, *IEEE Trans. Comput.*, Vol.53, No.3, pp.241–250 (2004).

[10] Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D. and Anderson, J.: Wireless Sensor Networks for Habitat Monitoring, *Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications* (*WSNA*), pp.88–97 (2002).

[11] Marcy, H.O. and Kaiser, W.J.: Wireless Integrated Network Sensors: Low power systems on a chip, *Proc. 24th European Solid State Circuits Conference*, pp.9–16 (1998).

[12] Nowak, R. and Mitra, U.: Boundary Estimation in Sensor Networks: Theory and Methods, *Proc. IPSN 2003*, Zhao, F. and Guibas, L. (Eds.), LNCS 2634, pp.80–95 (2003).

[13] Rahman, R., Alanyali, M. and Saligrama, V.: Distributed tracking in multihop sensor networks with communication delays, *IEEE Trans. Signal Processing*, Vol.55, pp.4656–4668 (2007).

[14] Rosencrantz, M., Gordon, G. and Thrun, S.: Decentralized sensor fusion with distributed particle filters, *Proc. Conference on Uncertainty in Artificial Intelligence* (*UAI*), pp.493–500 (2003).

[15] Shin, J., Guibas, L. and Zhao, F.: A Distributed Algorithm for Managing Multi-target Identities in Wireless Ad-hoc Sensor Networks, *Proc. IPSN 2003*, Zhao, F. and Guibas, L. (Eds.), LNCS 2634, pp.223–238 (2003).

[16] Shrivastava, N., Buragohain, C., Agrawal, D. and Suri, S.: Medians and Beyond: New aggregation techniques for sensor networks, *Proc. 2nd International Conference on Embedded Networked Sensor Systems*, pp.239–249 (2004).

[17] Skraba, P., Fang, Q., Nguyen, A. and Guibas, L.: Sweeps over wireless sensor networks, *Proc. 5th International Conference on Information Processing in Sensor Networks* (*IPSN*), pp.143–151 (2006).

[18] Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Bouonadonna, P., Gay, D. and Hong, W.: A macroscope in the redwoods, *Proc. 3rd ACM Conference on Embedded Networked Sensor Systems* (*SenSys*), pp.51–63 (2005).

[19] Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J. and Welsh, M.: Monitoring volcanic eruptions with a wireless sensor network, *Proc. 2nd European Workshop on Wireless Sensor Networks* (*EWSN*), pp.108–120 (2005).

[20] Zhu, X., Sarkar, R., Gao, J., Mitchell, J.S.B.: Light-Weight Contour Tracking in Wireless Sensor Networks, *Proc. INFOCOM 2008*, pp.1175–1183 (2003).

[21] Zhu, X., Sarkar, R. and Gao, J.: Segmenting a sensor field: Algorithms and applications in network design, *ACM Trans. Sensor Networks*, Vol.5, No.2 (2009).

Satoshi Fujita received the B.E. degree in electrical engineering, M.E. degree in systems engineering, and Dr.E. degree in information engineering from Hiroshima University in 1985, 1987, and 1990, respectively. He is a Professor at Faculty of Engineering, Hiroshima University. His research interests include communication algorithms on interconnection networks, parallel algorithms, graph algorithms, and parallel and distributed computer systems. He is a member of IEICE, SIAM Japan, IEEE, and SIAM.

Yang Yang received the B.M. degree in computer science (E-commerce) from Liaoning Normal University in China in 2005, and M.E. degree in information engineering from Hiroshima University in 2009. Her research interests include distributed networks and parallel algorithms. She is a system engineer of financial solution development department in Fujitsu Limited now.