EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Potential-based routing for supporting robust any-to-any communication in wireless sensor networks

Shinya Toyonaga[1*], Daichi Kominami[2], Masashi Sugano[3] and Masayuki Murata[1]

Abstract

In most applications, wireless sensor networks are supposed to operate in an unattended manner for a long period after sensor nodes' deployment. However, in such networks, sensor nodes frequently become faulty and unreliable because of the harsh environment of the observed area. Therefore, protocols used in wireless sensor networks must be designed to be robust. Moreover, because the battery capacity of a node is limited, energy savings are crucial in wireless sensor networks. To meet the requirements of future diverse wireless sensor networks, a sophisticated any-to-any routing protocol is thus required. As well as meeting the typical demands of wireless sensor networks, an any-to-any routing protocol needs to achieve low energy consumption, high scalability, robustness, and reliability. In this paper, we realize a potential-based any-to-any routing protocol (PBAR) by merging potential-based upstream and downstream routing. In PBAR, sensor nodes can send data to a certain sensor node by routing the data via a sink node. In simulation experiments, we show that, given a suitable node density, PBAR attains a data delivery ratio greater than 99.7%. We also show that the data delivery ratio recovers immediately after failure of 30% of sensor nodes or failure of a sink node.

Keywords: Sensor networks; Potential-based routing; Any-to-any routing; Simulation

1 Introduction

Wireless sensor networks have recently attracted attention as a fundamental technology of the Internet of Things, an energy management system, or an emergency system. The cost reduction of sensors has led to and is expected to continue to lead to an increase in the use of wireless sensor networks. In most applications, sensor nodes are supposed to operate in an unattended manner for a long period after their deployment. Owing to the harsh environment of wireless sensor networks and the large number of sensor nodes, it is not uncommon that sensor nodes become faulty and unreliable [1]. Therefore, it is essential that protocols used in wireless sensor networks are robust. In this paper, robustness means that the performance does not degrade even when a node fails. Moreover, energy savings are important in wireless sensor networks because sensor nodes are generally battery-powered.

Further expansion of future applications of wireless sensor networks will diversify the demands of the networks. To meet the requirements of future wireless sensor networks, a sophisticated any-to-any routing protocol needs to be realized. For an example of such applications, in-network processing such as reactive tasking, data querying, or data-centric storage requires communication between sensor nodes [2]. In addition to typical demands of wireless sensor networks, an any-to-any routing protocol needs to realize low energy consumption, high scalability, robustness, and reliability [3].

During the past few decades, various many-to-one upstream (sensor-to-sink) routing protocols have been studied since an upstream communication pattern and limited resources of sensors are assumed. Many potential-based routing protocols, which are one type of such routing protocols, aim for low overheads, high scalability, and energy balancing [4-7]. In potential-based upstream routing (PBUR), each node has a scalar value that is called its

*Correspondence: s-toyonaga@ist.osaka-u.ac.jp
[1]Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565-0871, Japan
Full list of author information is available at the end of the article

potential. Each node calculates its potential according to local information, such as the potentials or residual energy of the neighbor nodes. A sensor node whose hop count to a sink is smaller (larger) has a higher (lower) potential. Therefore, if a node sends data to its neighbor node with higher potential, the data will ultimately reach a sink node. Since these potential fields are constructed on the basis of purely local information, PBUR is highly scalable. PBUR is also robust because the potential field is changed dynamically and adapts to the condition of the network by updating potentials periodically. Moreover, if these potential fields are constructed according to residual energy, load balancing can be realized.

There is also a demand for downstream (sink-to-sensor) routing [8]. For example, a sink node sends a query to a specific sensor node upon receiving abnormal data from the sensor node, or a sink node sends a message to change the frequency of sensing in a specific domain.

To deliver downstream traffic, we proposed a potential-based downstream routing (PBDR) for multisink wireless sensor networks, which retains the advantage of potential-based routing [9]. We showed that PBDR realizes sink-to-sensor routing and meets the requirements of wireless sensor networks because of the adaptive behavior of the potential field.

In this paper, we realize a potential-based any-to-any routing (PBAR) protocol, which guarantees a high data delivery ratio and robustness against failure of nodes. Any-to-any routing is accomplished by merging PBUR and PBDR. The data flow of PBAR is shown in Figure 1. When each sensor node needs to send data to a certain node, the data are first delivered to a sink node by PBUR and then delivered to their destination node by PBDR.

We evaluate the data delivery ratio of PBAR at various node densities and path stretch, which is the ratio of the hop count of PBAR to the shortest hop count, to show the overhead of our protocol. We also evaluate our protocol's robustness against the failure of multiple sensor nodes or the failure of a sink node. Note that communication between arbitrary nodes can be realized without going through a sink node once both nodes retain each other's virtual coordinates.

The rest of the paper is organized as follows: We start by giving an overview of related work in Section 2. Sections 3 and 4 respectively show the potential-based upstream and downstream routing protocols. We present the proposed PBAR protocol in Section 5. In Section 6, we evaluate the performance of PBAR through simulation experiments. Finally, Section 7 gives our conclusions.

2 Related work

For wireless *ad hoc* sensor networks, various any-to-any routing protocols have been studied. In the flooding method and gossiping method, messages are relayed on the basis of broadcasts [10,11]. These methods suffer from a high number of redundant transmissions, particularly when a few nodes in a specific domain are the destinations.

Many studies have been conducted on reactive and proactive routing protocols [12,13]. In reactive protocols, each node constructs routes only in the case where communication is required. Power consumption can then be cut when communication is not needed. The delay time, however, is longer for reactive protocols because of their route discovery procedures. This means that reactive protocols are not appropriate for real-time applications. In proactive protocols, end-to-end delay is small. However, there is overhead because all the nodes collect information about links.

Geographic routing protocols allow for communication between two arbitrary nodes [14]. Equipment for acquiring the precise geographic position is required for these protocols, and all the nodes must know the position of their destinations. The virtual coordinate assignment protocol (VCap) is able to route data using a virtual position without the need for Global Positioning System devices [15]. In VCap, all nodes have three shortest hop counts from three anchor nodes and use them as virtual coordinates. Note that since the hop count is an integer, some nodes may have the same virtual coordinates in VCap.

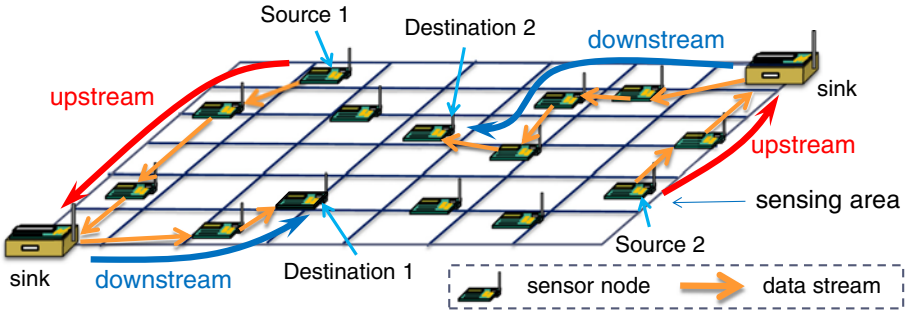

Figure 1 Data flow of PBAR via a sink node.

Many other virtual coordinate assignment protocols have been proposed [16-20].

PBUR protocols are categorized as being proactive. In PBUR, all the nodes have a scalar potential that constructs a potential field. Each node updates its potential according to local information, such as the potentials of its neighbors, its residual energy and that of its neighbors, or the hop count to a sink node. A sensor node whose hop count to a sink is smaller (larger) has a higher (lower) potential. Each node with data to be sent forwards the data to a node whose potential is higher than its own, and the data ultimately reach the sink node. Moreover, load balancing and extending the lifetime of wireless sensor networks using the residual energy of neighbor nodes or the amount of traffic have been studied [5,6,21].

In this paper, we propose any-to-any routing based on PBUR and PBDR. PBAR, like PBUR, is a type of proactive routing, and PBAR has scalability because it is realized through local information exchange only. Additionally, our method achieves a high data delivery ratio and robustness against failure of nodes. Better load balancing is also expected when a potential field considering a residual energy is constructed. In the simulation experiment carried out in this paper, a potential field constructed using our method is based on a controlled potential-based routing (CPBR [21]). However, our method is applicable to any strategy for potential field construction. We give details of PBUR in the following section.

3 Potential-based upstream routing protocols

Potential-based routing delivers data along the gradient of the potential field constructed over a wireless sensor network. A potential is a scalar value (like electric potential) and calculated by each sensor node from local interactions using the potential of its neighbors, its residual energy and that of its neighbors, or the hop count to a sink node. Smaller (larger) hop count from a sink node leads a higher (lower) potential to a sensor node. Thus, the gradient of a potential field means the direction to a sink node.

CPBR [21] constructs a potential field for multisink wireless sensor networks using the diffusion equation (Equation 1). The equation provides the magnitude ϕ of the diffusing quantity at time t and position X:

$$\frac{\partial \phi(X,t)}{\partial t} = D\Delta\phi(X,t), \tag{1}$$

where D is the diffusion rate and takes a positive value. By discretizing this equation and regarding ϕ as a potential, it is possible to construct a potential field from local information only.

A discrete form of the diffusion equation is described as Equation 2. $\phi(n,t)$ describes the potential of node n at time t. $Z(n)$ is a set of nodes neighboring node n. A parameter $D(n)$ changes the magnitudes of influences of the potentials of the neighbor nodes. It is noteworthy that potentials may oscillate when $D(n)$ is large. In CPBR, $D(n)$ is set to $\frac{\epsilon}{|Z(n)|}$ to keep the potential from oscillating, where $|Z(n)|$ is the cardinality of the set $Z(n)$. It is then considered that each node has been affected by the potential of essentially only one node. As a result, ϵ is set to a value between 0 and 1 to keep the potential from oscillating.

$$\phi(n,t+1) = \phi(n,t) + D(n) \sum_{k \in Z(n)} \{\phi(k,t) - \phi(n,t)\}. \tag{2}$$

Figure 2 shows the shape of the potential field after convergence. In CPBR, the potential of each sink node is calculated according to the number of data received by each sink node to realize load balancing. The potential of each sensor node deployed at the boundary of the network is set to the minimum potential so that the potentials of all nodes will not eventually arrive at a value much the same as the potential of the sink node. The potential of each of the other sensor nodes converges to the average value of the potentials of the neighbor nodes by using Equation 2. The potential field is then constructed, and monotonicity from a sensor node deployed at the boundary of the network to a sink node is ensured.

In existing PBUR protocols, there is a possibility that some sensor nodes have the same potential because PBUR guarantees uniqueness of only a potential assigned to a sink node. Therefore, when the sink node transmits data to a certain sensor node along the gradient of the potential field constructed through existing PBUR protocols, the data will not always arrive at the destination. This problem is treated as a contour problem, as shown in Figure 3. The contour problem means that no node can determine a next hop because no node knows the geographic direction to its destination node from the potentials.

We focused on the advantages of PBUR for wireless sensor networks and implement downstream routing by extending PBUR. The details of PBDR are given in the following section.

4 Potential-based downstream routing

PBDR, which we proposed in [9], must accomplish the following three tasks to handle the contour problem:

1. Assign potentials to all sensor nodes to identify them
2. Inform the sink nodes of the potentials
3. Route data to a destination node using its potential as a virtual location

In the following PBDR algorithm, we suppose that all sinks can communicate with each other via the wired link. In Section 4.1, we present the overview of our method. The node identification algorithm is then presented in Section 4.2. We show how to manage virtual coordinates of destination nodes in Section 4.3 and the downstream

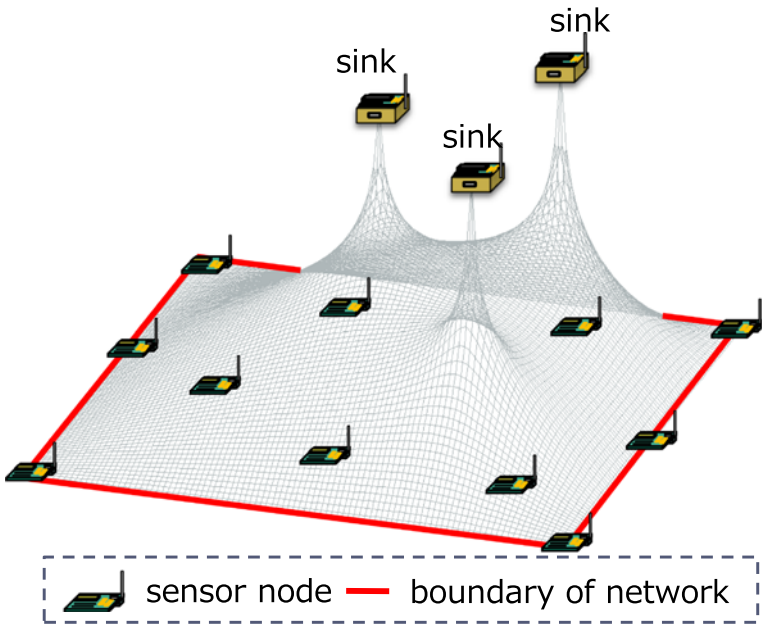

Figure 2 The shape of a potential field constructed by CPBR.

routing algorithm in Section 4.4. In our protocol, the local-minimum problem arises when a node cannot select a next hop. In Section 4.5, we explain this problem and how to solve it. Finally, the media access control (MAC) layer protocol we use is presented in Section 4.6.

4.1 Overview of PBDR

To realize PBDR, it is first necessary to assign potentials to all sensor nodes to identify them. We denote such a potential as P_{id}, and we give an overview of PBDR with P_{id} below:

1. Node identification: Each sensor node calculates its own P_{id}.
2. Management of P_{id}: When a sensor node generates an upstream data packet, it includes its P_{id} in the packet header. A sink node sends the upstream data to a system on the user's terminal when it receives the upstream data. The system on the user's terminal then records the P_{id} .
3. Downstream routing: We define a function $\mathrm{Dist}_p(n_1, n_2)$ that is a virtual distance between nodes n_1 and n_2 and is calculated from their P_{id}s. A sensor

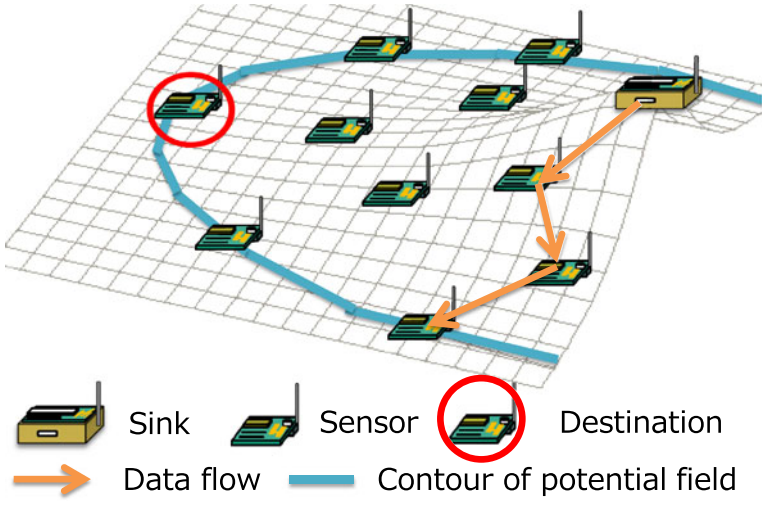

Figure 3 Contour problem for downstream routing using an existing potential construction method.

node with downstream data to be transmitted forwards the data to the neighbor node whose distance to the destination node is smallest, as shown by the value of function $\mathrm{Dist}_p(n_1, n_2)$. In this way, the data ultimately reach the destination node.

4. Local-minimum problem: A node cannot select a next hop node for downstream data when the local-minimum problem arises. We resolve the local-minimum problem by using another virtual distance function when the local-minimum problem occurs.

4.2 Node identification

In protocols based on existing methods for constructing a potential field, downstream data will not always arrive at the destination node because of the contour problem. Thus, we assign a virtual coordinate to all sensor nodes to identify them. This method is based on the idea of trilateration. N sink nodes individually construct potential fields, and all nodes have a set of potentials as a virtual coordinate. Here, as in [21], the diffusion equation is used by sink node i to construct the potential field F_i ($i = 1, \cdots, N$). We can now define that P_{id} is a set of N potentials and use P_{id} as a destination address. If there are at least three sink nodes and three potential fields, PBDR can be realized. In Section 6, we use four sink nodes and four potential fields to acquire redundancy in case a sink node fails.

Equation 3 is used to construct the potential field F_i having potential $\phi(n, t, i)$ at node n and time t. ϵ is a constant that plays the same role as ϵ in Equation 2.

$$\phi(n, t+1, i) = \phi(n, t, i) + \frac{\epsilon}{|Z(n)|} \sum_{k \in Z(n)} \{\phi(k, t, i) - \phi(n, t, i)\}. \tag{3}$$

Generally, in the diffusion equation, when all boundary conditions have the same value, all values in the field converge to the value of the boundary conditions, and the field eventually becomes flat. Consequently, potential routing does not work because there is no gradient in the field without a boundary condition. Therefore, we use Equation 4 as a boundary condition so that the potentials of the entire network do not converge to the potential of a sink node. S is a set of sink nodes. Note that sink node i constructs the potential field F_i.

$$\forall s \in S, \phi(s, t, i) = \begin{cases} \phi_{\max} & \text{if } i = s \\ \phi_{\min} & \text{otherwise.} \end{cases} \tag{4}$$

4.3 Management of P_{id}

As mentioned in Section 4.2, we use multiple sink nodes and multiple potential fields to acquire redundancy in case a sink node fails. We assume that all sink nodes are connected to the user's terminal via the wired link and can communicate with each other. When a sink node receives upstream data, it sends the data to the user's terminal, and a system on the user's terminal records the tuple {source node's ID, source node's P_{id}, generation time of the data} in its look-up table. When a user wants to send a query or a control message, the user commands the system to send downstream data and the system selects the sink node that is closest to the destination. The selected sink node then starts to send the downstream data.

We show the upstream and downstream packet formats in Tables 1 and 2, respectively. In upstream routing, a source node includes its P_{id} in a packet header, and sinks obtain the P_{id} when they receive the packet. In downstream routing, a sink includes P_{id} of the destination in a packet header, and relay nodes can refer to the P_{id} of the destination. The time to live (TTL) is the maximum number of hops that data can be forwarded. A loop flag for a downstream packet is used in checking whether the data is in a loop, and we explain how to use this in Section 4.5.

4.4 Downstream routing

As we have described in the previous section, we assume that a user's terminal and each sink node can communicate with each other via the wired link, and a user commands the system to send downstream data. A downstream data packet can then be routed to the sink node closest to a destination node, and the sink node can start delivery of the downstream data.

We define potential distance as the virtual distance calculated from P_{id}. To select a next hop, node n calculates the potential distance Dist_p between its neighbor $k (\in Z(n))$ and destination node d:

$$\mathrm{Dist}_p(k, d) = \sqrt{\sum_{i=1}^{N} (F_i(k) - F_i(d))^2}, \tag{5}$$

where $F_i(k)$ is the potential of node k in the potential field F_i, and $F_i(d)$ is the potential of destination node d. We use potential distance as a routing metric. A sink node includes P_{id} of destination node d in the header of a downstream data packet, and relay nodes forward the data to node n_1 that fulfills the following condition:

$$n_1 = \arg \min_{k \in Z(n)} \mathrm{Dist}_p(k, d). \tag{6}$$

For the sake of greater reliability, our downstream routing allows a sender node to forward data to the neighbor

Table 1 Upstream packet format

Source node ID (2 byte)	Destination node ID (2 byte)
Sender node ID (2 byte)	Receiver node ID (2 byte)
Sequence number (2 byte)	TTL (1 byte)
P_{id} (source) (16 byte)	
Sensing data (72 byte)	

Table 2 Downstream packet format

Source node ID (2 byte)	Destination node ID (2 byte)	
Sender node ID (2 byte)	Receiver node ID (2 byte)	
Sequence number (2 byte)	TTL (1 byte)	loop flag (1 byte)
P_{id} (destination) (16 byte)		
Query or control message (72 byte)		

that is the second closest to the destination node. Two neighbor nodes n_1 and n_2 are then candidates for the next hop node. In Section 4.6, we explain how to select the next hop node from n_1 and n_2 in detail. When data reach a neighbor node of the destination node, they are forwarded to the destination node using a node ID.

4.5 Local-minimum problem

In the local-minimum problem, a sender node has no neighbor node whose Dist_p is smaller than that of the sender. This problem occurs around void areas, as is well known in geographic routing [22]. Once the problem arises, sender nodes cannot forward data.

In the example shown in Figure 4, node C must forward data to node D so that the destination node receives the data. However, Dist_p(node D, destination) is larger than Dist_p(node B, destination), and node C does not forward data to node D. We use a local detour rule, by which node v forwards data to node w having the smallest $\mathrm{Dist}_p(w, \mathrm{destination})$, even if $\mathrm{Dist}_p(v, \mathrm{destination})$ is smaller than $\mathrm{Dist}_p(w, \mathrm{destination})$, and node v does not forward data to node u after node v receives the data from node u. According to this rule, node C forwards the data to node B. As a result, a data packet will follow a loop through node A, node B, and node C.

The local-minimum problem occurs when a destination node is near the boundary of the monitoring area. This is because the node density near the boundary of the monitoring area is low, which leads to a void area. Hence, we assume that a destination node exists near the boundary of the monitoring area when a loop is detected. We then resolve the local-minimum problem using an alternative routing metric.

The main idea to solve this problem is using only one potential field when a loop is detected. Because the diffusion equation is a harmonic function, a loop rarely occurs when a single potential field is used for downstream routing. The node near the monitoring area boundary is located in the area farthest from a certain sink node, and the potential of the destination node in the potential field built by the sink node is nearly equivalent to $\phi_{\min}$ by using Equation 3. Thus, the possibility that the data packet gets close to the boundary of the monitoring area is high when a node forwards the packet to the node farthest from the sink node. To send data to the boundary of the monitoring area, we use the potential field whose potential is the smallest in P_{id} of the destination node. From the above, we define a potential gap $\mathrm{Gap}(k, d)$ (Equation 7) and use it as an alternative routing metric when a routing loop is detected. $\mathrm{Gap}(k, d)$ is a potential distance between node k and destination d on the potential field, F_i, where the potential of the destination is the smallest among all $F_j (1 \leq j \leq N)$. Then, a node which detects a loop calculates potential gaps of its neighbors and forwards data to the neighbor whose potential gap is the smallest among them.

$$\mathrm{Gap}(k, d) = |F_i(k) - F_i(d)|\,,\ i = \arg\min_{1 \leq j \leq N} F_j(d). \quad (7)$$

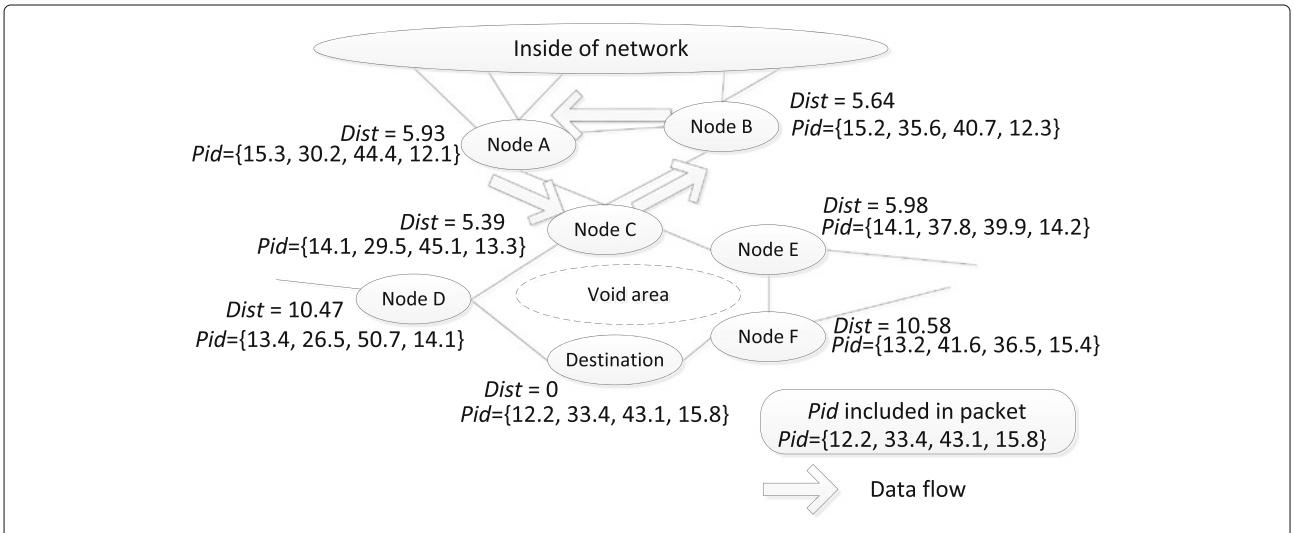

Figure 4 Local-minimum problem.

For example, in Figure 4, the potential gap of node A is 3.1, that of node C is 1.9, and that of node D is 1.2. Node C then forwards the data to node D, and the data reach the destination node.

A sequence number and a loop flag are included in the data packet header and are used to detect routing loops. When a node receives a downstream data packet, the node records the sequence number of the data. When a node receives data with the same sequence number, the node judges that a loop has occurred and sets the loop flag to one. Each node records n_{hist} sequence numbers of received packets from the newest received one.

In addition to the case where Dist_p is used, when Gap is used, our downstream routing allows a sender node to forward data to the neighbor that is the second closest to the destination node. In Algorithm 1, we show how to determine two candidates for a next hop node. How to select a next hop node from those two nodes is shown in detail in the following section.

Algorithm 1 Select a next hop for downstream routing

Require: Node n receives a data packet whose destination is node d

Ensure: Return two candidates for a next hop node that have the smallest or second smallest $Dist_p$ or Gap to the destination

if loop_flag of the received data packet equals to one **then**
  $next\_hop.address1 \Leftarrow getNextHopUsingGap(d, n)$
  $next\_hop.gap1 \Leftarrow getGap(d, next\_hop.address1)$
  $next\_hop.address2 \Leftarrow getSecondNextHopUsingGap(d, n)$
  $next\_hop.gap2 \Leftarrow getSecondGap(d, next\_hop.address2)$
  if $Gap(n, d) < next\_hop.gap1$ **then**
    loop_flag of the received data packet is set to zero
    $next\_hop.address1 \Leftarrow getNextHopUsingDist_p(d, n)$
    $next\_hop.dist1 \Leftarrow getDist_p(d, next\_hop.address1)$
    $next\_hop.address2 \Leftarrow getSecondNextHopUsingDist_p(d, n)$
    $next\_hop.dist2 \Leftarrow getSecondDist_p(d, next\_hop.address2)$
  end if
else
  if sequence number of the received data packet is in the set of node n's history **then**
    loop_flag of the received data packet is set to one
    $next\_hop.address1 \Leftarrow getNextHopUsingGap(d, n)$
    $next\_hop.gap1 \Leftarrow getGap(d, next\_hop.address1)$
    $next\_hop.address2 \Leftarrow getSecondNextHopUsingGap(d, n)$
    $next\_hop.gap2 \Leftarrow getSecondGap(d, next\_hop.address2)$
  else
    $next\_hop.address1 \Leftarrow getNextHopUsingDist_p(d, n)$
    $next\_hop.dist1 \Leftarrow getDist_p(d, next\_hop.address1)$
    $next\_hop.address2 \Leftarrow getSecondNextHopUsingDist_p(d, n)$
    $next\_hop.dist2 \Leftarrow getSecondDist_p(d, next\_hop.address2)$
  end if
end if
return *next_hop*

4.6 MAC layer protocol

In this paper, we use intermittent receiver-driven transmission (IRDT) for the MAC layer protocol [23]. In IRDT, all nodes sleep and wake up asynchronously with the duty cycle T_{duty}. Whenever a node wakes up, it sends an ID message that informs neighbor nodes that the node is ready to receive data.

When node n_s forwards data to node n_r in IRDT, the procedure shown in Figure 5 is used. Node n_s with data to be sent wakes up and waits for an ID message from node n_r. Upon receiving an ID message from node n_r, node n_s sends an SREQ message, which is a communication request informing node n_r that it has a data packet for node n_r. When node n_r receives the SREQ message, it stays awake and sends to node n_s a RACK message, which is an acknowledgement of the communication request. Afterward, node n_s sends a data message to node n_r. Finally, node n_r sends to node n_s a DACK message, which is an acknowledgement of the data. If node n_r is not a destination node, node n_r becomes a sender and waits for an ID message from a neighbor node.

Node n_s drops the data when forwarding the data does not succeed within T_{timeout} after node n_s wakes up. Additionally, when the number of forwardings exceeds the TTL, node n_s drops the data.

In our protocol, there are two candidates for a next hop, n_1 and n_2, for obtaining reliability. n_1 is the first closest node and n_2 is the second closest node to the destination. Here, we explain how to select a next hop node from n_1 and n_2. When a sender node n_s receives an ID message from n_1 or n_2, n_s stochastically determines whether or not it returns an SREQ message to the sender of the ID regarding it as a next hop node. When Dist_p is used for a potential distance metric, the probability of selecting n_1 is 1, and that of selecting n_2 is $\frac{\text{Dist}_p(n_s,d)+\text{Dist}_p(n_1,d)}{\text{Dist}_p(n_s,d)+\text{Dist}_p(n_2,d)}$.

The probability of selecting n_2 is close to 1 when the difference between $\text{Dist}_p(n_1, d)$ and $\text{Dist}_p(n_2, d)$ is small. In such a case, both nodes n_1 and n_2 are suitable to be the next hop because the distance to the destination node is almost the same. We add $\text{Dist}_p(n_s, d)$ to the numerator and denominator so as to provide multipath even when $\text{Dist}_p(n_1, d)$ is almost zero. Similarly, when Gap is used, the probability of selecting n_1 is 1 and that of selecting n_2 is $\frac{\text{Gap}(n_s,d)+\text{Gap}(n_1,d)}{\text{Gap}(n_s,d)+\text{Gap}(n_2,d)}$.

5 Potential-based any-to-any routing

5.1 Outline

As mentioned in Section 1, we realize PBAR by merging PBUR and PBDR. We assume that each sensor node only has its own and its neighbors' P_{id}s, and each sink node has all sensor nodes' P_{id}s. This means that each sensor node does not have the destination node's P_{id} when generating data for a certain node. Therefore, in PBAR, the data

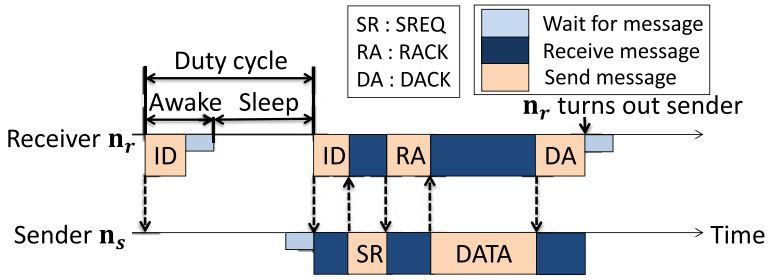

Figure 5 Procedure of forwarding data in IRDT.

are delivered to a sink node first, and the sink node then includes the destination's P_{id} in the header of the data. The following is an outline of PBAR:

1. A source node sends data to a sink node through PBUR when it generates data for a certain sensor node.
2. When a sink node receives the data, it sends the data to the sink node that is closest to the destination node in terms of potential distance via the wired link. The sink node then starts to send the data to the destination node through PBDR.

5.2 Any-to-any routing

First, each sink node constructs its own potential field, and all nodes have a set of potentials as a virtual coordinate. After convergence of P_{id}s, each sensor node generates upstream data packets containing its P_{id} and forwards them to a sink node through PBUR. When a sink node receives the data, the node broadcasts the data to the other sink nodes via the wired link, and all sink nodes can record the P_{id}s of all sensor nodes. Afterward, when a sensor node generates data destined for a certain sensor node, the data are delivered to the closest sink node through PBUR.

Note that not all data have to go through a sink node. When the destination node is a neighbor node of one of the relay nodes, the relay node can send the data to the destination node directly.

In the method described above, much data has to go through one of the sinks, and this may result in large path stretch. However, in some cases, the number of hops until the data arrival at the destination node may be less than that of the shortest hop path. Figure 6 shows one example of such cases. In this example, the wired link between sink nodes can be used as a shortcut link. When data generated by the source node are forwarded along the shortest hop path, they go through nodes A, B, and C before arriving at the destination node. Here, the number of hops is four. In PBAR, the source node first forwards data to sink A. Sink A then sends the data to sink B via the wired link because sink B is the closest sink to the destination node. Finally, sink B forwards the data to the destination. Therefore, the number of hops is two for wireless communication and one for wired communication. Because there

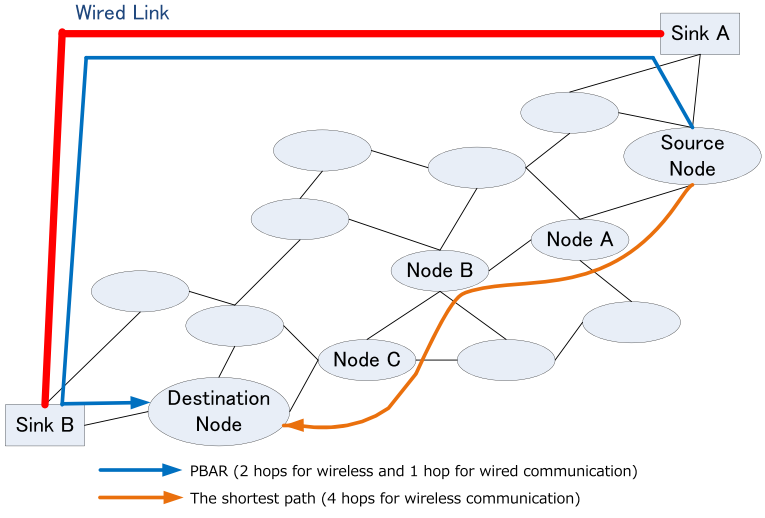

Figure 6 An example showing smaller hop number in PBAR than in the shortest hop routing.

are fewer wireless communications between sensor nodes, sensor nodes can save their energy.

5.3 Constraints

In our proposed method, at least three sink nodes must be deployed in the sensing area. To identify each sensor node, a virtual coordinate needs to be assigned to a sensor node uniquely. When less than three sink nodes are deployed and each of them constructs its own potential field, some sensor nodes may have the same virtual coordinate because our method is based on a theory of triangulation. In such a case, data may be routed to a sensor node that is not a destination node.

To send data to a specific sensor node, sink nodes must know the P_{id} of a destination node. Therefore, each sensor node needs to send data to a sink node periodically. In this paper, each sensor node uses the potential field with the highest potential at its P_{id} to send upstream data to the nearest sink node. A source node inserts its P_{id} into the header of the packet, and sink nodes can collect P_{id} for each sensor node. Each sink node discards a P_{id} when it does not receive a new one from a sensor node for $T_{\mathrm{expiration}}$. A data packet is dropped when a sink node tries to send the packet to a destination node but no sink node has P_{id} of the destination node. This means that the sink nodes temporarily could not receive any upstream data from the destination node. Such a packet drop occurs when an upstream data packet is dropped, when the delay is large, or when a destination node is isolated.

To realize PBAR, deployed sink nodes need to be connected to each other with a high-speed link. Otherwise, sink nodes could not share all P_{id}s and downstream data could not be routed to the sink node that is closest to the destination node.

In our proposed method, many data may be dropped because of congestion. When node A has data to be forwarded to node B and node B has data to be forwarded to node A at the same time, neither node can forward the data. This leads to small throughput or to the dropping of data due to timeout. Hence, many data may be dropped when traffic is heavy and congested. This influence becomes remarkable around the sink nodes because all data go through a sink node. When the queue of a sink node is full with downstream data, a neighbor node of the sink node cannot forward upstream data to the sink node, and vice versa. Thus, many nodes around the sink node cannot forward data unless some data are dropped. Therefore, our proposed method targets the situation of comparatively low traffic load. However, this constraint can be loosened easily by enlarging the queue size of nodes.

6 Simulation experiments

In this section, we present the results of our simulation experiments. PBAR is implemented on the OMNeT++ [24] network simulator. We evaluate our method in two situations: one where all the data go through a sink node and the other where a relay node of which the destination node is a neighbor sends the data to the destination node directly. We denote the former as situation 1 (S1) and the latter as situation 2 (S2). We evaluate the data delivery ratio and the path stretch of PBAR at various node densities in S1 and S2.

Table 3 Simulation configuration

Parameter	Radio range	TTL	Data size	Other message size	Bandwidth
Value	100 m	30	100 byte	28 byte	100 kbps
	$\phi_{\max}$	$\phi_{\min}$	ϵ	n_{hist}	
	90	0	0.8	3	
	T_{duty}	T_{update}	T_{timeout}	$T_{\mathrm{expiration}}$	
	1 s	100 s	5 s	2,500 s	

The sensor nodes are randomly distributed in a 600 m $\times$ 600 m square. In this network, the number of deployed sensor nodes is from 50 to 250, and four sink nodes are situated at the four corners of the observation area. The rate of data generation is $\frac{1}{300}$ per sensor node for any-to-any communication in a Poisson process. The model of radio attenuation is the free-space model [25], and we assumed that no noise exists. Each sensor node selects a destination node randomly and starts to send the data to the destination node through PBAR when it generates the data. The queue size of each node is one, and a node with data does not broadcast an ID message. The other parameter settings are summarized in Table 3. Under these conditions, we evaluate how the data delivery ratio is affected by node density (Section 6.1), by sensor node failure (Section 6.2), and by sink node failure (Section 6.3).

6.1 Data delivery ratio

Simulation results for S1 are shown in Tables 4 and 5. The number of trials is 50, and the confidence interval is 95%. The simulation time is 30,000 s. In those tables, Drop_{TTL},

Table 4 Data delivery/drop ratio for S1

Number of sensor nodes	Node density	Data delivery ratio (%)	$\mathrm{Drop}_{\mathrm{TTL}}$ (%)	$\mathrm{Drop}_{\mathrm{timeout}}$ (%)	$\mathrm{Drop}_{\mathrm{no\_info}}$ (%)
50	Lower	99.33 ± 0.19	0.584	0.053	0.023
100	Lower	99.67 ± 0.07	0.203	0.098	0.021
150	Medium	99.77 ± 0.02	0.016	0.183	0.025
200	Higher	99.07 ± 0.09	0.007	0.894	0.023
250	Higher	97.65 ± 0.09	0.007	2.312	0.024

Table 5 Path stretch in S1

Number of sensor nodes	Path stretch
50	2.88
100	3.53
150	3.40
200	3.78
250	4.01

$\mathrm{Drop}_{timeout}$, and $\mathrm{Drop}_{no_{info}}$ mean the packet drop ratio when the number of forwarding of data exceeds the TTL, when a node with a data packet cannot forward the data within T_{timeout} after the node generates or receives it, and when no sink node has P_{id} of the destination node, respectively.

The data delivery ratio is low when the node density is low because there are few links in the entire network and the local-minimum problem thus easily occurs. This is clear from the fact that Drop_{TTL} is comparatively high when the number of sensor nodes is 50 or 100. The data delivery ratio is high when the node density is high because there are more links in the entire network. When the node density is excessively high, however, packet collisions and congestion occur frequently, especially near sink nodes, thus decreasing the data delivery ratio. This is shown by the fact that $\mathrm{Drop}_{timeout}$ is comparatively high when the number of sensor nodes is 200 or 250. The data delivery ratio is highest when the number of nodes is 150. In that case, the data delivery ratio is 99.7% and the average number of neighbor nodes is 16.7.

As shown in Table 5, the average number of hops is approximately three to four times that of the shortest hop path. This is because all data go through a sink node. However, the path stretch can be decreased when a relay node that is a neighbor of the destination node sends the data to the destination node directly.

Simulation results for S2 are shown in Tables 6 and 7. As shown in Table 6, the data delivery ratio and the drop ratio have the same characteristic as those shown in Table 4. The data delivery ratio is low when the node density is low or excessively high. However, in comparison with Table 4, $\mathrm{Drop}_{timeout}$ is lower and $\mathrm{Drop}_{no_{info}}$ is higher. The reason for the former is that the traffic load around sink nodes is low because not all the data have to go through a sink node, and the reason for the latter is that sink nodes may not receive the data and update the sensor nodes' P_{id}s. When a sink node cannot update the sensor nodes' P_{id}s for a long time, it discards the P_{id}s.

Table 6 Data delivery/drop ratio for S2

Number of sensor nodes	Node density	Data delivery ratio (%)	$\mathrm{Drop}_{\mathrm{TTL}}$ (%)	$\mathrm{Drop}_{\mathrm{timeout}}$ (%)	$\mathrm{Drop}_{\mathrm{no\_info}}$ (%)
50	Lower	99.34 ± 0.16	0.485	0.058	0.116
100	Lower	99.58 ± 0.07	0.175	0.073	0.165
150	Medium	99.70 ± 0.02	0.017	0.161	0.109
200	Higher	99.20 ± 0.05	0.005	0.656	0.128
250	Higher	98.19 ± 0.07	0.005	1.665	0.135

Table 7 Path stretch in S2

Number of sensor nodes	Path stretch
50	2.32
100	2.74
150	2.78
200	2.99
250	3.17

Comparing Table 7 with Table 5, the path stretch is smaller in S2. The average number of hop is as much as three times that of the shortest hop path. Note that the path stretch can be about one once the source node retains the destination node's virtual coordinate. This is because relay nodes can calculate the potential distance and find the closest next hop to the destination node without data going through a sink node when the source node includes the destination node's virtual coordinate in the header of the data.

Because of an imperfect routing table, PBAR cannot guarantee the data delivery ratio to be 100%. However, an improvement is possible. A larger queue size that allows a node to forward data quickly may reduce $\mathrm{Drop}_{\mathrm{timeout}}$. $\mathrm{Drop}_{\mathrm{no_{info}}}$ can be alleviated when each node sends upstream data that contains its P_{id} to a sink node periodically. Although $\mathrm{Drop}_{\mathrm{TTL}}$ can hardly be reduced owing to the local-minimum problem, other next hop decision strategies or the retransmission of the upper-layer protocol allows a high data delivery ratio.

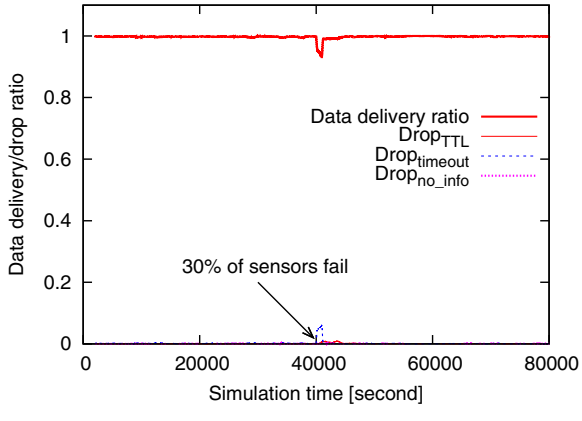

Figure 7 Data delivery/drop ratio in the case where sensor nodes fail.

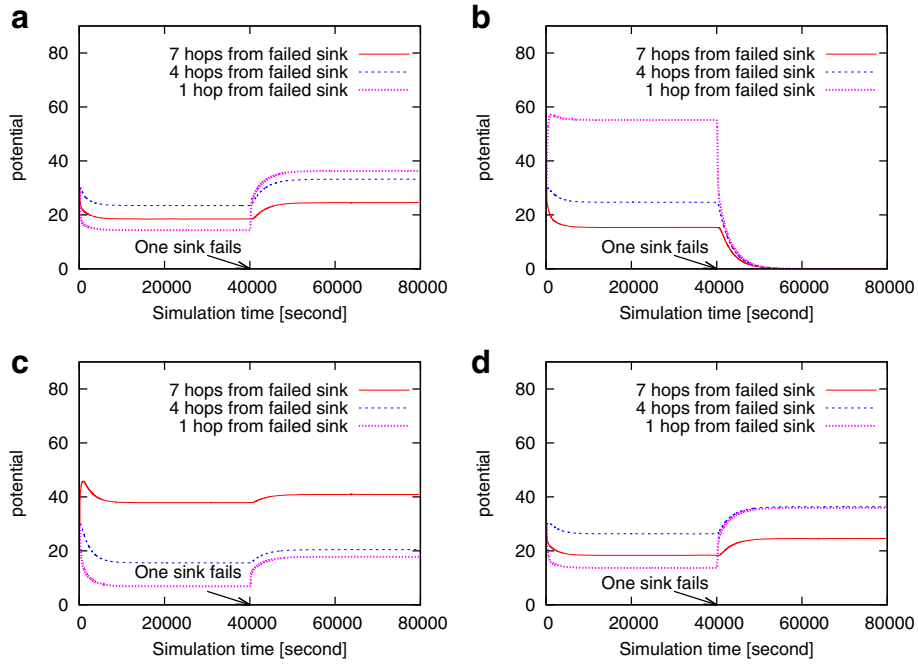

Figure 8 Potential convergence after sink node failure. (a) Potential F_0, **(b)** potential F_1, **(c)** potential F_2, and **(d)** potential F_3.

6.2 Failure of sensor nodes

In the case where 150 nodes and 45 sensor nodes fail, we evaluate the data delivery ratio from t - 1,000 (s) to t (s) at each time t. The simulation time is 80,000 s, and 45 sensor nodes fail after 40,000 s have elapsed. The number of trials is 10. We assume that the situation is S1.

Figure 7 shows the data delivery ratio and the drop ratio from t - 1,000 (s) to t (s) at each time t. The results show that our proposed routing works well even if sensor nodes fail. The data delivery ratio decreases steeply when sensors fail at 40,000 s but quickly returns to the level observed before the failure of nodes. $\mathrm{Drop}_{\mathrm{TTL}}$ and $\mathrm{Drop}_{\mathrm{no_{info}}}$ do not change considerably when sensor nodes fail, but $\mathrm{Drop}_{\mathrm{timeout}}$ increases steeply. This is because a sensor node cannot select the next hop that is closer to the destination node until the potential fields converge.

From this result, PBAR is robust against the failure of sensor nodes. Even after 30% of sensor nodes fail, the data delivery ratio is more than 93.1%. It takes about 1,200 s for the data delivery ratio to recover to 99% after the sensor nodes fail, which relates to the period in which each node updates its potential. Therefore, when each sensor node updates its potential more frequently, it takes less time to recover the data delivery ratio, while the energy consumption increases.

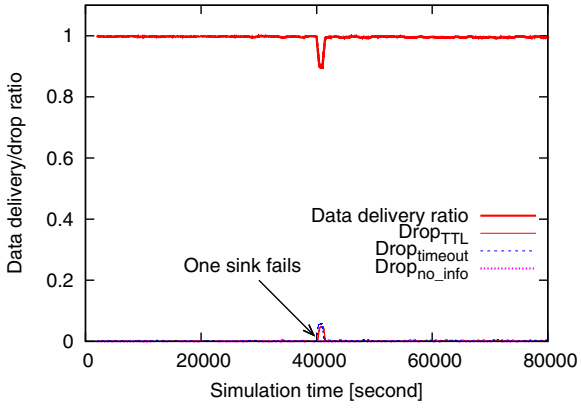

Figure 9 Data delivery/drop ratio in the case that a sink node fails.

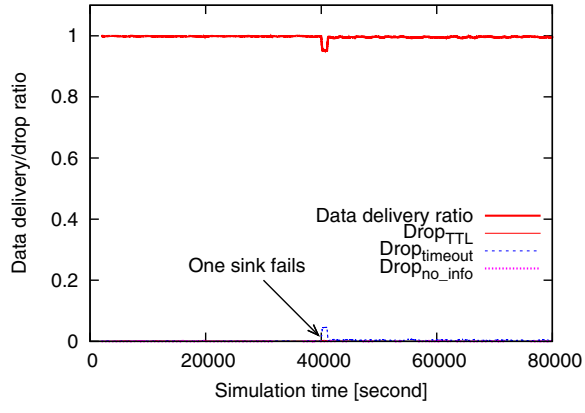

Figure 10 Data delivery/drop ratio in the case that a sink node fails in S3.

6.3 Failure of a sink node

In the case where 150 nodes and one of four sink nodes fail, we evaluate the data delivery ratio immediately prior to 1,000 s. The simulation model is the same as that for sensor node failures. When sink node s fails, all the potentials in F_s converge on $\phi_{\min}$ because of the boundary condition (4). A sensor node with upstream data to be sent decides the next hop according to the potential field whose value is highest among the potentials. In this manner, the other three sink nodes collect the P_{id} for each sensor node and PBAR regains its effectiveness after the sink node failure.

Figure 8 shows changes in potential until the potential fields converge. Here, the changes in potential for three nodes are shown. The first node is the farthest from the failed sink node, with a hop count of seven to the failed sink node. The second is deployed near the center of the network, with a hop count of four to the failed sink node. The third is a one-hop neighbor of the failed sink node. In Figure 8b, the changes in the potential field constructed by the failed sink node are shown and the potentials converge to $\phi_{\min}$ (= 0) in about 30,000 s. In Figure 8a,c,d, the changes in the potential fields that the other three sink nodes construct are shown and the potentials converge in about 20,000 s.

The data delivery ratio and the drop ratio from t - 1,000 (s) to t (s) at each time t are shown in Figure 9. In Figure 9, the data delivery ratio decreases steeply when one of the sink nodes fails at 40,000 s but quickly recovers to the level observed before the failure. $\mathrm{Drop}_{\mathrm{no_{info}}}$ does not change considerably when a sink node fails, but $\mathrm{Drop}_{\mathrm{TTL}}$ and $\mathrm{Drop}_{\mathrm{timeout}}$ increase steeply.

The reason why $\mathrm{Drop}_{\mathrm{timeout}}$ and $\mathrm{Drop}_{\mathrm{TTL}}$ increase is that the sensor node that is deployed around the failed sink node keeps sending data in upstream routing to the failed sink node until potential fields converge. When a neighbor node of the failed sink node has data in upstream routing, the node waits for an ID message from the failed sink node. Because the failed sink node cannot send an ID message, the neighbor node of the failed sink node drops the data owing to timeout.

In our method, a node updates its potential when the node receives an ID message containing a potential. Therefore, a node with data that is awake for a long time updates its potential more frequently. When a sensor node that has upstream data waits for an ID message from the failed sink node, its potential may become less than that of its neighbor node owing to frequent updates of potential. It then forwards the data to the neighbor sensor node. When the data are forwarded many times, similarly, the data are finally dropped due to the expiry of the TTL.

The result shows that PBAR is robust against failure of a sink node. It takes about 1,500 s for the data delivery ratio to recover to 99% after one of the sink nodes fails, which relates to the time when the potential field constructed by the failed sink is no longer used. Therefore, when each sensor node updates its potential more frequently or when a neighbor node of the failed sink node detects the failure of the sink node and broadcasts a message to no longer use the potential field constructed by the sink node, it takes less time for the data delivery ratio to recover. We denote the latter situation as situation 3 (S3) and evaluate the data delivery ratio immediately prior to 1,000 s.

Figure 10 shows the result for S3. The transition of the data delivery and drop ratio is similar to the result in Figure 9. However, in S3, Drop_{TTL} does not increase even when a sink node fails. This is because all the upstream data are delivered to one of the three other sink nodes after sensor nodes receive information about the failure of a sink node. The time for the data delivery ratio to recover to 99% after the failure of one sink node decreases to 1,100 s.

Note that the time for the data delivery ratio to recover is much less than the time of potential convergence. This is because it is not the potential convergence itself but the gradient of the potential field that is important in potential-based routing. Therefore, the time for potential convergence does not greatly affect the data delivery ratio.

7 Conclusions

In this paper, we realize PBAR by merging PBUR and PBDR. In PBAR, multiple sink nodes construct independent potential fields, and all nodes have a set of potentials used as a virtual coordinate. We defined virtual distance based on virtual coordinates and use it as a routing metric. Through OMNeT++ simulation, we evaluated the data delivery ratio and path stretch for various node densities, as well as the robustness against failure of multiple sensor nodes or a sink node. PBAR achieves a data delivery ratio greater than 99.7% when the network has a suitable node density. Even if multiple sensor nodes fail or a sink node fails, the data delivery ratio recovers immediately after sensor node failure or sink node failure.

In PBAR, when the number of potential fields increases, the reliability of any-to-any routing increases, but so does the overhead. We plan to investigate this trade-off in future work.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This research was supported in part by 'Grants-in-Aid for Scientific Research (C) (23500097)' from the Japan Society for the Promotion of Science (JSPS).

Author details
[1]Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565-0871, Japan. [2]Graduate School of Economics, Osaka University, Toyonaka, Osaka 560-0043, Japan. [3]School of Knowledge and Information Systems, College of Sustainable System Sciences, Osaka Prefecture University, Sakai, Osaka 599-8531, Japan.

Received: 13 June 2013 Accepted: 7 November 2013
Published: 5 December 2013

References

1. SA Quadri, O Sidek, Software maintenance of deployed wireless sensor nodes for structural health monitoring systems. Int. J. Comput. Engineer. Sci. **3**(2), 1 (2013)
2. R Fonseca, S Ratnasamy, J Zhao, CT Ee, D Culler, S Shenker, I Stoica, Beacon vector routing: scalable point-to-point routing in wireless sensornets, in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2, NSDI'05* (USENIX Association, Berkeley, 2005), pp. 329–342
3. S Duquennoy, O Landsiedel, T Voigt, Let the tree Bloom: scalable opportunistic routing with ORPL, in *Proceedings of ACM Conference on Embedded Networked Sensor Systems, SenSys '13* (ACM, New York, 2013), pp. 2:1–2:14
4. C Wu, R Yuan, H Zhou, A novel load balanced and lifetime maximization routing protocol in wireless sensor networks, in *Proceeding of IEEE Vehicular Technology Conference* (IEEE, Singapore, 2008), pp. 113–117
5. Q Liang, D Yao, S Deng, S Gong, J Zhu, Potential field based routing to support QoS in WSN. J. Comput. Inform. Syst. **8**(6), 2375–2385 (2012)
6. PTA Quang, DS Kim, Enhancing real-time delivery of gradient routing for industrial wireless sensor networks. IEEE Trans. Industrial Informatics. **8**(1), 61 (2012). doi:10.1109/TII.2011.2174249
7. H Yoo, M Shim, D Kim, A scalable multi-sink gradient-based routing protocol for traffic load balancing. EURASIP. J. Wireless Commun. Netw. **2011**(1), 1 (2011). doi:10.1186/1687-1499-2011-85
8. SJ Park, R Sivakumar, IF Akyildiz, R Vedantham, GARUDA: Achieving effective reliability for downstream communication in wireless sensor networks. IEEE Trans. Mobile Comput. **7**(2), 214 (2008)
9. S Toyonaga, D Kominami, M Sugano, M Murata, Potential-based downstream routing for wireless sensor networks, in *Proceedings of International Conference on Systems and Networks Communications (ICSNC 2012)* (IARIA, Lisbon, 2012), pp. 59–64
10. S Guo, Y Gu, B Jiang, T He, Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links, in *Proceedings of ACM MobiCom* (ACM, Beijing, 2009), pp. 133–144
11. S Fauji, K Kalpakis, A gossip-based energy efficient protocol for robust in-network aggregation in wireless sensor networks, in *Proceedings of IEEE Pervasive Computing and Communications Workshops* (IEEE, Seattle, 2011), pp. 166–171
12. T Clausen, P Jacquet, C Adjih, A Laouiti, P Minet, P Muhlethaler, A Qayyum, L Viennot, INRIA, Optimized link state routing protocol. Internet Draft draft-ietf-manet-olsr-txt Work in progress, 1–53 (2004)
13. CE Perkins, EM Royer, Ad-hoc On-Demand Distance Vector Routing, in *Proceedings of IEEE Mobile Computing Systems and Applications* (IEEE, New Orleans, 1999), pp. 90–100
14. L Shu, Y Zhang, L Yang, Y Wang, M Hauswirth, N Xiong, TPGF: geographic routing in wireless multimedia sensor networks. Telecommun. Syst. **44**, 79–95 (2010)
15. A Caruso, S Chessa, S De, A Urpi, GPS free coordinate assignment and routing in wireless sensor networks, in *Proceedings of IEEE INFOCOM 2005* (IEEE, Miami, 2005), pp. 150–160
16. MJ Tsai, HY Yang, BH Liu, WQ Huang, Virtual-coordinate-based delivery-guaranteed routing protocol in wireless sensor networks. IEEE/ACM Trans. Netw. **17**(4), 1228 (2009). doi:10.1109/TNET.2008.2008002
17. J Zhou, Y Chen, B Leong, B Feng, Practical Virtual Coordinates for large wireless sensor networks, in *Proceedings of IEEE International Conference on Network Protocols (ICNP 2010)* (IEEE, Kyoto, 2010), pp. 41–51
18. S Xia, X Yin, H Wu, M Jin, XD Gu, Deterministic greedy routing with guaranteed delivery in 3D wireless sensor networks, in *Proceedings of ACM MobiHoc 2011* (ACM, Paris, 2011), pp. 1:1–1:10. doi:10.1145/2107502.2107504
19. M Caesar, M Castro, EB Nightingale, G O'Shea, A Rowstron, Virtual ring routing: network routing inspired by DHTs. SIGCOMM Comput. Commun. Rev. **36**(4), 351 (2006)
20. A Awad, R German, F Dressler, Exploiting virtual coordinates for improved routing performance in sensor networks. IEEE Trans. on Mobile Comput. **10**(9), 1214 (2011)
21. D Kominami, M Sugano, M Murata, T Hatauchi, Controlled potential-based routing for large-scale wireless sensor networks, in *Proceedings of Modeling, Analysis and Simulation of Wireless and Mobile Systems* (ACM, New York, 2011), pp. 187–196
22. Y Noh, U Lee, P Wang, BSC Choi, M Gerla, VAPR: Void aware pressure routing for underwater sensor networks. IEEE Trans. on Mobile Comput. **12**(5), 895–908 (2012). doi:10.1109/TMC.2012.53
23. C Damdinsuren, D Kominami, M Sugano, M Murata, T Hatauchi, Lifetime extension based on residual energy for receiver-driven multi-hop wireless network. Cluster Computing (2012). doi: 10.1007/s10586-012-0212-0
24. A Varga, *Modeling and Tools for Network Simulation* (Springer, Berlin, 2010), pp. 35–59
25. R Nagel, S Eichler, Efficient and realistic mobility and channel modeling for VANET scenarios using OMNeT++ and INET-framework, in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops* (ICST, Marseille, 2008), pp. 89:1–89:8

doi:10.1186/1687-1499-2013-278
Cite this article as: Toyonaga *et al.*: **Potential-based routing for supporting robust any-to-any communication in wireless sensor networks.** *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:278.