Data Dissemination for Wireless Sensor Networks

Min-Gu Lee
Network Control Platform Development Team
BcN Business Unit, KT Corporation
bluehope@kt.co.kr

Sunggu Lee
Division of EECE
POSTECH, Pohang, Kyungbuk, S. Korea
slee@postech.ac.kr

Abstract

Due to the special characteristics (limited battery power, limited computing capability, low bandwidth, need to collect sensor data from multiple fixed-location source nodes to a sink node that may be mobile, etc.) of wireless sensor networks, routing algorithms designed for general mobile ad hoc networks may not be directly applicable to wireless sensor networks. In one possible routing scheme for wireless sensor networks, each node maintains up-to-date hop-distances and next-hop nodes to the mobile sink node (or multiple mobile sink nodes). However, this type of method may require too much control overhead in order to maintain up-to-date and consistent hop-distances and next-hop nodes for all of the sensor nodes in the network. Therefore, we propose a new low-control-overhead data dissemination scheme, referred to as pseudo-distance data dissemination*, for efficiently disseminating data packets from all sensor nodes to mobile sink nodes in a wireless sensor network.*

1 Introduction

Wireless sensor networks(WSN) are composed of a massive number of cheap battery-powered sensor nodes with wireless communication capability. They sense environmental information such as humidity, temperature, sound, light and motion, and special nodes referred to as sink nodes collect the sensed data to produce useful analysis outputs. The performance of a WSN is limited by battery power, low computing capacity, short wireless transmission range and hostile environments. Therefore, routing schemes for WSN should be carefully designed to ensure long battery life, low computing requirements, low control overhead and efficient communications.

One of the major differences between WSN and general mobile ad hoc networks(MANETs) is in the communication patterns used. In MANETs, routing is performed for each source to destination pair such that any node in a MANET can be a destination as well as a source node. However, in a WSN, only one, or at most a few, sink nodes can be destinations that gather sensed data. In addition, the main communication pattern in a WSN is *from multiple sources to one (or a few) destination*, and those multiple sources repetitively transmit data packets to one (or a few) mobile sink node. Therefore, there can be a lot of static source nodes per mobile sink node in a WSN. Since routing algorithms that are commonly used in MANETs try to provide one or more paths for each source to destination pair in a peer-to-peer manner, the number of control messages required rapidly increases as the number of sources per destination is increased. However, since exchanging messages consumes battery power (typically, transmitting power is at least 20 times the power required in an idle state), exchanges of control messages should be suppressed in order to save battery power, in addition to scarce bandwidth. Therefore, a new routing scheme that disseminates sensed data from potentially all sensor nodes to one (or at most a few) mobile sink nodes, while taking into consideration all of the above mentioned considerations, would be highly beneficial for WSNs.

Another major difference between WSN and MANETs concerns the mobility of nodes. All nodes in the MANETs are potentially mobile while most of nodes are not mobile in WSN. It is commonly assumed that deployed sensor nodes do not have mobility in WSN. However, sinks can be mobile in many application scenarios of WSN. Therefore routing scheme of WSN should support mobile sinks. Note that in many applications such as tactical operations or environmental sensing applications, it is assumed that sink nodes have enough battery and computing power because sink nodes should be powerful devices with large battery capacity or they can be equipped to vehicles.

2 Related Works

A data centric routing scheme called *directed diffusion* [1] was proposed to support data dissemination in WSN. In directed diffusion, a new communication

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

paradigm for WSN was proposed that a sink node floods *interests* messages and sensor nodes that have matched data send them to the corresponding sink node. Since interests and other control messages are generated periodically, directed diffusion can keep updating paths to sink nodes even if some links fail. However, mobility of sink nodes can rapidly degrade performance of directed diffusion since path can not updated until next interest is flooded through the whole network. In order to catch topological changes caused by mobility of sink nodes, frequent flooding of interests is required but it introduces too much overheads of bandwidth and battery consumption of sensor nodes. Furthermore, since exploratory data is following all possible paths in the network following gradients, directed diffusion generates too much unnecessary communications overhead.

Another data dissemination scheme for WSN is two-tier data dissemination(TTDD) [2]. TTDD disseminates data using location information via external devices such as GPS that considers mobility of sink nodes. Although TTDD may efficiently support mobile sink nodes, it requires location information which is difficult to achieve in WSN since GPS is expensive and GPS consumes battery power to detect its location. In addition, TTDD can not be used in some geographical regions like forests or inside of caves since GPS requires direct line-of-sight to the satellites.

3 Pseudo-Distance Data Dissemination (PDDD)

Since each sensor node in WSN is potentially a source node to the mobile sink node, all sensor nodes in the network should maintain routes to the mobile sink node. Although mobility of sensor nodes is restricted in WSN, data dissemination schemes for WSN should be able to follow topological changes.For instance, a network topology can be changed if animals kick sensor nodes, enemy troops destroy sensor nodes, or battery of sensor nodes are exhausted, etc. In addition, in order to save battery and bandwidth, any kind of periodical messages should be avoided. Therefore, data dissemination schemes for WSN should be *reactive*.

3.1 Assumptions and Notation

In this paper, it is assumed that all links are bidirectional and no control messages are lost. Broadcasting network is assumed since most off-the-shelf wireless networking devices are omni-directional(Zigbee, IEEE 802.11, Bluetooth, etc.). It is also assumed that each nodes in the network has its own unique identifier (like a MAC address). Mobile sink nodes are assumed that have unlimited battery power, so we do not take care about the battery efficiency of sink nodes. Finally, network partitioning is not considered in this paper.

A network is modeled as an undirected graph $G = (V, E)$, where V is a finite set of nodes and E is a set of bidirectional communication links at a given time instant. A sensor node with unique identifier i is denoted as v_i and a neighbor set of node v_i, written as N_i, consists of all nodes that have a bidirectional link to node $v_i \in V$.

3.2 Data Dissemination

We adapt the data dissemination scenario from the directed diffusion. When a mobile sink node v_{sinkID} wants data from sensor nodes, it broadcasts an *interests* message to its neighbor sensor nodes N_{sinkID} with `type`, `duration`, and `interval` information. Each sensor node v_i that receives an interest message floods the received interest message to its own neighbor sensor nodes. If v_i has matched data to the `type` in the interest message, then it begins to transmit data packets at every specified time `interval` for a specified time `duration` in the interest message. A mobile sink periodically broadcasts interest messages for active tasks.However, sensor nodes do not send exploratory data and do not wait reinforcement messages because each sensor node already has valid routes to the sink node. Note that directed diffusion has to send exploratory data until it receives reinforcement containing a selected route by the sink node. Readers should refer to [1] for more details about data dissemination.

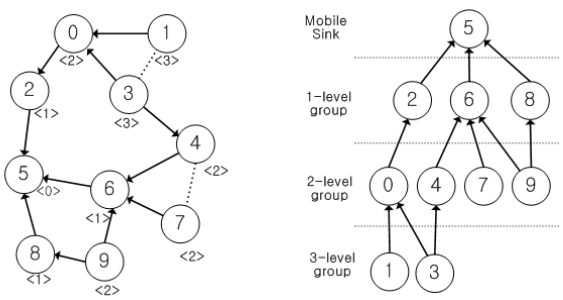
3.3 Partially Ordered Graph using Hop-Distance

If all nodes in the network know their own hop-distances to the sink node and hop-distances of their own neighbors, then optimal routes can be achieved in terms of path lengths by partially ordered set of V using hop-distances. A simple assignment of parent-child relationships between adjacent nodes as a parent for lower hop-distance node and a child for a greater hop-distance node builds a *partially ordered graph*(POG). Then, by forwarding packets to a parent node, optimal data dissemination is achieved in terms of path length until network topology is unchanged.

Figure 1 shows an example of POG using hop-distances and its hierarchical view. The numbers in the vertices of Fig. 1 represent unique ID of each node and the numbers beside vertices are hop-distance of each node toward sink node v_5. Directed solid edges represent parent-child relationships between adjacent nodes, and undirected dotted edges represent undefined links between adjacent nodes. It is trivial to see that all sensor nodes have at least one path to the sink node v_5 if vertices are ordered by hop-distances.

Suppose that by movements of sink node v_5, link $e_{6,5}$ is broken. Then, v_6 can not determine its correct hop-distance until it receives an interest message. Therefore, mobile sink

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE



(a) Partially ordered graph (b) Hierarchical view

Figure 1. An example of partially ordered graph and its hierarchical view.

nodes should re-floods a new interest message whenever topology of network is changed in order to assign up-to-date and consistent hop-distances for all sensor nodes. However, it is not feasible because detection of topological changes at a sink node is very difficult. Furthermore, frequent flooding of interest messages introduces too much overheads.

Another problem of POG is wasting of links among sibling nodes. For instance, a link $e_{1,3}$ can be used to disseminate data packets from v_1 to v_5 using a path $(e_{1,3}, e_{3,4}, e_{4,6}, e_{6,5})$ if links on a path $(e_{1,0}, e_{0,2}, e_{2,5})$ are highly congested or some of links are failed. However, there is no order between sibling nodes v_1 and v_3, v_1 can not disseminate data packets to the sink node until it receives interest message from the sink node if some of links on a path $(e_{1,0}, e_{0,2}, e_{2,5})$ are broken. Therefore we claim that *totally ordered graph*(TOG) that siblings also have orders can achieve better performance than POG. In addition, the proposed algorithm adopts *pseudo-distance* concept from [3], because ordered graph using hop-distances is difficult to catch up topological changes.

3.4 Level Assignments

PDDD(pseudo-distance data dissemination) assigns a level to each sensor node for a corresponding sink node with pseudo-distance. The level of a node v_i to a sink node v_{sinkID} is written as $L_{i,sinkID} =< \lambda, -\alpha, -\beta, v_i >$. A pseudo-distance $\lambda_{i,sinkID}$ is a distance metric between a sensor node v_i and a sink node v_{sinkID}. α is the number of neighbors that have lower λ values than v_i and β is the number of neighbors that have the same λ value. Finally, v_i represents the unique ID of the node. Nodes $v_i \in V$ are ordered by lexicographical comparisons of level of each node $L_{i,j}$. For more details, readers should refer [3].

PDDD sets parent-child relationships between adjacent nodes using pseudo-distance of each node. A node v_i becomes a parent node of v_j if there exists a direct link $e_{i,j} \in E$ and $\lambda_{i,sinkID} > \lambda_{j,sinkID}$. Consequently, v_j is denoted as a child of v_i.Nodes with same pseudo-distance values are grouped into a level group. Then, in order to forward packets toward the sink node, each sensor node simply selects a parent node with the minimum pseudo-distance as its next hop. Thus, all neighbors with the smallest λ value are considered first. Among adjacent nodes with same pseudo-distance values, a node v_k becomes an elder sibling node of v_l if there exists a direct link $e_{k,l} \in E$ and $\lambda_{k,sinkID} = \lambda_{l,sinkID}$ and $L_{k,sinkID} < L_{l,sinkID}$. In order to reduce the number of control messages required, temporarily incorrect α and β values are permitted since small deviations in the number of alternative subpaths are not catastrophic. Each node selects its next-hop as a neighbor with minimum level metric among parent and elder sibling neighbor nodes.

3.5 Totally Ordered Graph using Pseudo-Distance

In order to overcome shortcomings of POG, PDDD builds a TOG using pseudo-distance. In PDDD, sink nodes have the lowest level as $< 0, 0, 0, v_{sinkID} >$ and other sensor nodes have greater levels than the corresponding sink node. Initially, each sensor node v_i sets its level toward the sink node v_{sinkID} as $< \infty, \infty, \infty, v_i >$. When a sink node wants to collect data from sensor nodes for active tasks, it broadcasts an interest message. By receiving interest messages, each node can set its pseudo-distance and corresponding level, then it broadcasts the received interest message to its neighbor nodes with its own level metric. An interest message consists of six-tuple as [`sequence, sinkID`, L, `type, duration, interval`] where `sequence` is the sequence number of the interest message that is generated by the corresponding sink node, `sinkID` is the unique ID of the corresponding sink node, L is the level of the node that is broadcasting the interest message, `type` represents data type that the sink node want to collect, `duration` represents how long the data should be transmitted from sensor nodes that have matched data and `interval` represents the transmitting period of sensed data.

Figure 2 shows a pseudocode of the procedure executed when a node v_i receives an interest message from its neighbor node. On receiving an interest message, a sensor node v_i updates its neighbors' level metric table with specified L in the message (line 2 of Fig. 2). If v_i is the sink node that generates the interest message, then it has nothing to do. Therefore it simply drops that interest packet and terminates the procedure(lines 3–5 in Fig. 2).

When a sensor node v_i receives an interest message directly from a sink node v_{sinkID}, it initiates a local timer to the sink node v_{sinkID} with `MaxHeartBeatTime` to de-

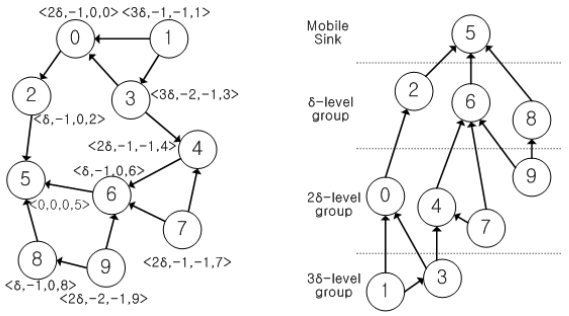
Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

```
1.  recvInterest(message p) {
2.      updateNeighborsLevel(p);
3.      if(p.sinkID = v_i) {
4.        drop(p);
5.        return;
6.      } else if (p.sinkID = v_j) {
7.        createTimer(p.sinkID);
8.        setTimer(MaxHeartBeatTime,p.sinkID);
9.      }
10.     if (p.sequence > sequence[sinkID]) {
11.       sequence[sinkID] = p.sequence;
12.       λ_{i,sinkID} = p.L.λ + δ;
13.       updateLevel();
14.       forwardInterest(p);
15.       if (hasMatchedData(p.type) {
16.         sendData(p.duration, p.interval);
17.       }
18.     } else if (p.sequence = sequence[sinkID]) {
19.       if (λ_{i,sinkID} − p.L.λ > δ) {
20.         λ_{i,sinkID} = p.L.λ + δ;
21.         updateLevel();
22.         forwardInterest(p);
23.       }
24.     }
25.     drop(p);
26.     return;
27. }
```

Figure 2. A pseudocode of the procedure that is executed when v_i receives an interest message from its neighbor node.

tect breakage of link $e_{i,sinkID}$. Since mobile sink nodes have enough battery power, only they generate periodical heart-beat messages to their direct neighbors. Therefore, direct neighbors of mobile sink nodes can detect link breakage to the sink nodes by loss of heartbeat messages. Each direct neighbor sensor nodes create a timer for the sink node (line 7 of Fig. 2). Note that the 'createTimer()' function does not create multiple timer instances for one sink node, i.e., only one timer is created for one mobile sink node. Then, it sets its timer as `MaxHeartBeatTime` (line 8 of Fig. 2).

Then, v_i checks the sequence number of the interest message(line 10 of Fig. 2). If the sequence number is newer than it has received, it should update its pseudo-distance as $\lambda_{i,sinkID} = \lambda_{j,sinkID} + \delta$ and corresponding level information including α and β (lines 13–14 of Fig. 2) after it updates its new sequence number(line 11 of Fig. 2) because a new interest message contains up-to-date level information.



(a) Totally ordered graph (b) Hierarchical view

Figure 3. An example of totally ordered graph and its hierarchical view.

After updating level metric, it forwards the interest message with its own level information $L_{i,sinkID}$ (line 14 of Fig. 2). Then v_i begins to send data packets toward the corresponding sink node if it has matched data during `p.duration` with a period of `p.interval` that are specified in the interest message (lines 15–17 of Fig. 2).

If the interest message has the same sequence number that v_i received last, then it compares its pseudo-distance to that of v_j. If the difference meets $\lambda_{i,sinkID} - \lambda_{j,sinkID} > \delta$, then v_i updates its pseudo-distance $\lambda_{i,sinkID} = \lambda_{j,sinkID} + \delta$ and the corresponding level information since the route following v_j is expected shorter than v_i has (lines 19–21 of Fig. 2). Then, v_i forwards the interest message with its own level information (line 22 of Fig. 2). Note that PDDD simply drops an interest message which is older than it already received.

Figure 3 shows an example of totally ordered graph with a mobile sink node v_5 using pseudo-distances. Numbers beside vertices represents a level metric of each node. As described above, a sink node v_5 sets its level as $< 0, 0, 0, 5 >$ and others have non-zero levels corresponding to its own pseudo-distance value. Note that the pseudo-distance between two nodes is a multiple of δ, which is the default difference in λ between adjacent nodes. δ is used for inserting a new level to already created levels. Details of inserting a new level using δ will be described in the next section. Since an interest message is generated by the sink node v_5, all sensor nodes can set their own levels $L_{i,5}$ as direct proportional to hop-distances to the sink node.

Nodes are also grouped into four different level groups as same as in Fig. 1(b). However, compared to Fig. 1(a), there is no undefined links in Fig. 3(a) since all nodes are totally ordered with level metrics $L_{i,5}$ that can create hierarchy between siblings in Fig. 3(b). Therefore, two undirected dotted edges in Fig. 1(a) becomes directed solid edges between siblings that can increase redundancies of routes.

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

3.6 Maintenance of TOG

Sink nodes are mobile and sensor nodes would be failed to operate due to various reasons such as exhausted battery power, out of order by hostile environments, etc. Therefore, PDDD should be able to dynamically and locally maintain routes without a new interest message from the sink node. Because there are redundant paths in PDDD, certain level of link failures can be tolerated in PDDD. However, if a node v_i loses its all parent nodes and elder sibling nodes, then it has to update its own level locally to find new parent nodes.

Loss of parents are detected in two methods: missing heart-beat messages at direct neighbor nodes of a sink node or missing ACKs from next hop nodes at sensor nodes. As described in Sect. 3.5, a sink node periodically broadcasts heart-beat messages in order to notify link connectivity to its neighbors. If a direct neighbor node of sink node failed to receive consequent heart-beat messages from its sink node until the timer is expired, it regards the link as broken and it destroys the timer. For general sensor nodes, they waits ACK packet from its next hop after it transmits data packets. If it failed to receive ACK from its next hop, then it regards the link is broken.

Figure 4 shows the pseudocode of the procedure executed when a node v_i loses all of lower level neighbor nodes. When v_i loses its all lower level nodes, it checks whether it has at least one sibling neighbor node v_j, which is a node with same pseudo-distance $\lambda_{j,sinkID}$ and at least one child neighbor node v_k, which is a node with greater pseudo-distance $\lambda_{k,sinkID}$ (line 2 of Fig. 4). If v_i has both sibling neighbor nodes and child neighbor nodes, then it defines a new level group as $\lambda_{i,sinkID} = \lfloor(\lambda_{i,sinkID} + min_{v_j \in N_i}(\lambda_{j,sinkID}))/2\rfloor$ in order to set parent-child relationships between v_i and sibling neighbor nodes v_j (line 3 of Fig. 4). By assigning a new level group, it can transparently get new parent neighbor nodes without any affects to its child nodes. Otherwise, it sets its pseudo-distance as $\lambda_{i,sinkID} = min_{v_j \in N_i}(\lambda_{j,sinkID}) + \delta$ (line 5 of Fig. 4) to convert neighbors that has minimum pseudo-distance as parents. After updating corresponding pseudo-distance, v_i updates its own level metric $L_{i,sinkID}$ and broadcasts a new UPD message to notify changes of pseudo-distance to its neighbors. An UPD consists of [`originator`, `sequence`, $L_{i,sinkID}$, v_{sinkID}] where `originator` represents the node that detects topological changes, `sequence` represents the sequence number of generated UPD message at `originator` to distinguish fresh UPD messages, $L_{i,sinkID}$ represents a level of a node that transmitting the UPD message, and v_{sinkID} is an ID of the corresponding sink node.

Figure 5 shows a pseudocode of the procedure executed when a node v_i receives an UPD message from its neighbor node. First, v_i updates level information of neighbors

```
1.  lostAllParentsOrElderSiblings() {
2.      if(sibling neighbors & child neighbors exist) {
3.          λ_{i,sinkID} = ⌊(λ_{i,sinkID} + min_{v_j∈N_i}(λ_{j,sinkID}))/2⌋,
                where λ_{i,sinkID} < λ_{j,sinkID};
4.      } else {
5.          λ_{i,sinkID} = min_{v_j∈N_i}(λ_{j,sinkID}) + δ;
6.      }
7.      updateLevel();
8.      sendUPD(L_{i,sinkID});
9.      return;
10. }
```

Figure 4. Pseudocode of procedure executed when a node v_i loses all of its parent neighbor nodes and elder sibling neighbor nodes.

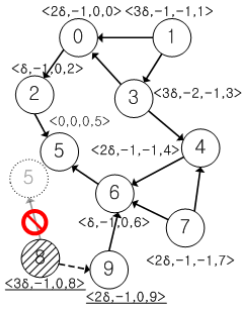
in its local table following received UPD message (line 2 of Fig. 5). Next, if pseudo-distance of v_i is ∞, then it updates its pseudo-distance and corresponding level metric, then broadcasts an UPD message with its own level information (lines 3–7 in Fig. 5). Note that `sequence`, `originator` information is copied from the received UPD message. Although pseudo-distance of v_i is not ∞, it may need to updates its pseudo-distance if it can find parent neighbor nodes that are expected to be shorter than it has. If $\lambda_{i,sinkID} - \mathrm{p}.L.\lambda$ is greater than δ, then it updates its pseudo-distance as $\lambda_{i,sinkID} = \mathrm{p}.L.\lambda + \delta$ and corresponding level metric (lines 8–10 in Fig. 5). After updating level metric, it broadcasts an UPD message with its own level information (lines 11–12 in Fig. 5).. Note that, as a previous case, `sequence, originator` information is copied from the received UPD message since the UPD message is triggered by other UPD messages. Finally, if a pseudo-distance of received message is greater than it has, which may cause loss of parent neighbor nodes or elder sibling neighbor nodes, it checks that it still has at least one valid parent of elder sibling neighbor nodes(lines 13–14 of Fig. 5). If it has neither a parent neighbor node nor a sibling neighbor node, then it has to execute the procedure "lostAllParentsOrElderSiblings()" (lines 15–16 of Fig. 5).

Figure 6 shows an example of route maintenance that a link $e_{8,5}$ is broken due to movement of a sink node v_5. Dashed arrows represents adjusted links due to topological changes. Since link $e_{8,5}$ is broken, v_8 can not receives heart-beat messages from v_5. After timer is expired, v_8 considers that it loses its parent neighbor node. Because it has neither parent neighbor nodes nor elder sibling neighbor node, it executes the procedure of Fig. 4. v_8 has to update its pseudo-distance as $\lambda_{8,5} = \lambda_{9,sinkID} + \delta$ because it does

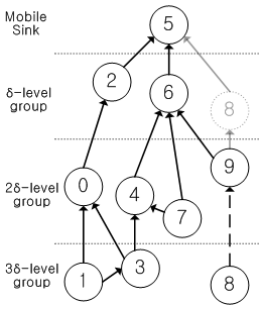
Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

```
1.  recvUPD(message p) {
2.      updateNeighbor(p);
3.      if (λ_{i,sinkID} = ∞) {
4.          λ_{i,sinkID} = p.L.λ + δ;
5.          updateLevel();
6.          sendUPD(L_{i,sinkID}, p);
7.          return;
8.      } else if (λ_{i,sinkID} − p.L.λ > δ) {
9.          λ_{i,sinkID} = p.L.λ + δ;
10.         updateLevel();
11.         sendUPD(H_{i,sinkID}, p);
12.         return;
13.     } else if (λ_{i,sinkID} ≤ p.L.λ) {
14.         if(isLostAllParentsOrElderSiblings()) {
15.             lostAllParentsOrElderSiblings(); {
16.             return;
17.         }
18.     }
19. }
```

Figure 5. A pseudocode of the procedure that is executed when v_i receives an UPD message from its neighbor node.

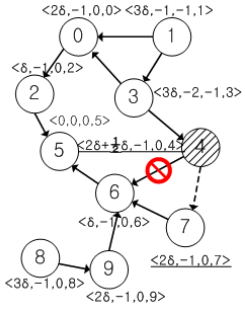


(a) A link $e_{8,5}$ is broken.

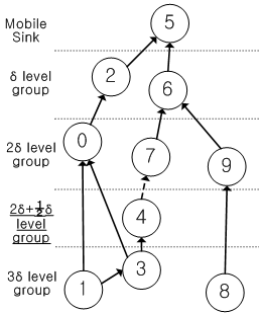


(b) Hierarchical view

Figure 6. An example with broken link $e_{8,5}$.



(a) A link $e_{4,6}$ is broken.



(b) Hierarchical view

Figure 7. An example with broken link $e_{4,6}$.

not have any child neighbor node (line 5 of Fig. 4). After updating λ, it updates corresponding level metric, then it broadcasts a new UPD message to its neighbors in order to notify the change of its level metric. On receiving an UPD message from v_8, v_9 updates its neighbors level information table (line 2 of Fig. 5). Since $p.L.\lambda$ is greater than $\lambda_{9,5}$, it checks whether it still has parent neighbor nodes or elder sibling neighbor nodes. As a result, v_9 still has one parent neighbor node.Note that v_9 does not broadcast an UPD message although it updates its level since its pseudo-distance is not changed as described earlier. Hierarchical view of Fig. 6(a) is shown in Fig. 6(b). v_8 is now in a level group of 3δ, which was δ group previously.

Figure 7 shows another example of route maintenance that link $e_{4,6}$ is broken. Since v_4 loses its last parent neighbor node and it does not have any elder sibling neighbor node, it executes the procedure of Fig. 4. As a result, v_4 defines a new level group as $\lambda_{4,5} = \lfloor(\lambda_{4,5} + min_{v_j \in N_4}(\lambda_{j,5}))/2\rfloor = 2\delta + 1/2\delta$ (lines 2–3 in Fig. 4). Note that by assigning a new level to v_4, v_3 is not affected that v_3 still can disseminate data packets via v_4 without any process. On receiving an UPD message from v_4, v_3 and v_7 update their corresponding α and β values. Although their level metrics are changed, they do not broadcast UPD messages since pseudo-distance is still unchanged.

4 Evaluation

Simulations were conducted to evaluate the proposed dissemination algorithm using ns-2, which is a discrete event simulator tool commonly used in networking research. Performance of TTDD is not directly comparable to PDDD since TTDD requires external devices that provide location information such as GPS. Therefore we compared performance of PDDD to directed diffusion(DD) only.

4.1 Simulation Environment

The distributed coordination function (DCF) of IEEE 802.11 for wireless LANs is used for the MAC and PHY layers. The data rate is set to 2Mbps and communication range is set to 250 units. The simulation space is a 2500 $\times$ 2500 unit^2 area. Power dissipation of idle time is 35mW, receive power dissipation is 395mW and transmit power dissipation is 660mW. Initial energy level of each node is 100 joule to ensure that battery of sensor nodes are not exhausted. The period of interest message is 5 seconds and each source node transmits 6 times of 64 bytes data packets at every second. The number of nodes including mobile sink nodes is 250 which is varies from 150 to 300. Num-

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

ber of mobile sink nodes varies from 1 to 10. Mobile sink nodes move with maximum speed from 0 to 20 units/s. Simulation was performed 111 seconds but data are collected after 11 seconds from the simulation begins in order to exclude initial unstabilities of each data dissemination algorithm. Therefore, effective simulation time is 100 seconds.

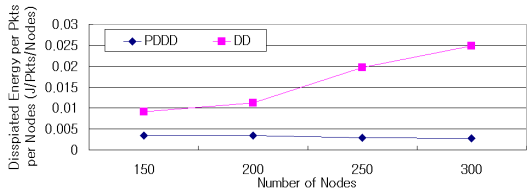
We choose three metrics to show performances of data dissemination algorithms: average dissipated energy per a successfully received data packets at a sink node per a node, average latency, and average distinct data packet delivery ratio. The first metric, the average dissipated energy per a successfully received data packets at a sink node per a node, shows the average dissipated energy by a node in delivering useful information to the sink nodes. The second metric, average latency, shows average one-way latency between transmitted time at source nodes and receiving at a sink node. The third metric, distinct-event delivery ratio, is fraction of the number of distinct events received at a sink node to the number of events originally sent at source nodes.

4.2 Simulation Results versus Number of Nodes

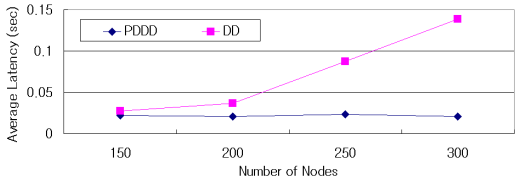
Figure 8 shows various performances versus number of nodes. The first graph, Figure 8(a) shows average dissipated energy. As the network size is increased, energy dissipation of directed diffusion is rapidly increased while PDDD remains almost constant. Because directed diffusion delivers exploratory data following gradients, it requires multiple transmissions of exploratory data. On the other hand, no exploratory data is required in PDDD and no multiple transmission is required in PDDD. Therefore PDDD is much more scalable than directed diffusion.

Figure 8(b) shows the average latency.In directed diffusion, average latency is also rapidly increased as the number of nodes is increased while it is remaining almost as a constant in PDDD.

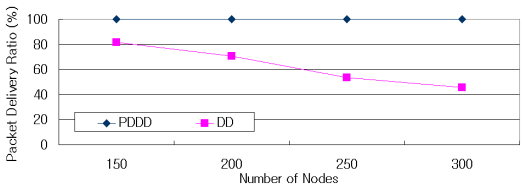
Figure 8(c) shows the average packet delivery ratio. As the number of sensor nodes is increased, the packet delivery ratio of directed diffusion is rapidly decreased while PDDD delivers almost all packets to the sink nodes.



(a) Average dissipated energy per a received packet per a node



(b) Average latency



(c) Packet delivery ratio

Figure 8. Performances vs. number of nodes.

4.3 Simulation Results versus Mobility of Sink Nodes

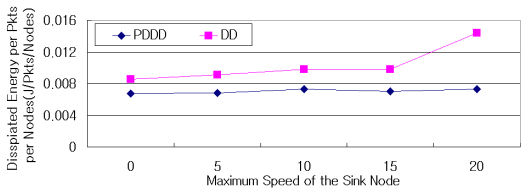
Figure 9 shows various performances versus mobility of sink nodes to see how sink mobility affects performance of dissemination algorithms. Figure 9(a) shows average dissipated energy.As the mobility of sink nodes is increased average dissipated energy is also increased. As depicted, energy dissipation of PDDD is much less than directed diffusion, and the gap is increased as the mobility of sink node is increased. Figure 9(b) shows the average latency.As mobility of sink nodes increased, latency should be increased because packets are waiting in queue at intermediate nodes during maintenance.Therefore latency should be increased as mobility of sink node increased. As depicted, PDDD achieves less latency than directed diffusion.

Figure 9(c) shows the average packet delivery ratio.As mobility of sink nodes increased, packet delivery ratio is getting worse. For low mobility of sink nodes, packet delivery ratio of directed diffusion is similar to PDDD since interest message is periodically retransmitted. However, packet delivery ratio is rapidly decreased in directed diffusion when mobility of sink nodes is very high because periodical flooding of interest message can not catch up topological changes while packet delivery ratio of PDDD is only slightly decreased.

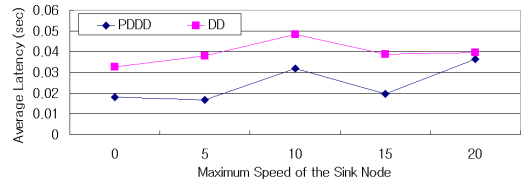
4.4 Simulation Results versus Number of Source Nodes per a Sink Node

Figure 10 shows various performances versus the number of source nodes per a sink node. Among them, Figure 10(a) shows average dissipated energy.As the number of source nodes increased, dissipated energy is decreased in PDDD while rapidly increased in directed diffusion. Note that multiple sources share route information for a sink node in PDDD. Therefore as the number of source nodes increased, control messages per a node is decreased. There-

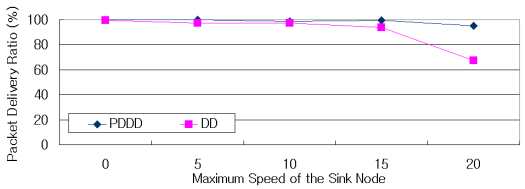
Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE



(a) Average dissipated energy per a received packet per a node

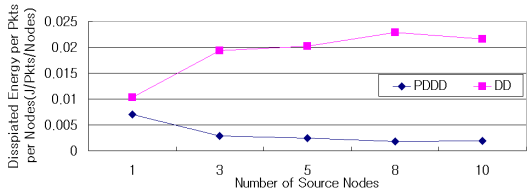


(b) Average latency

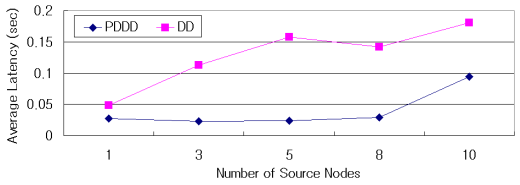


(c) Packet delivery ratio

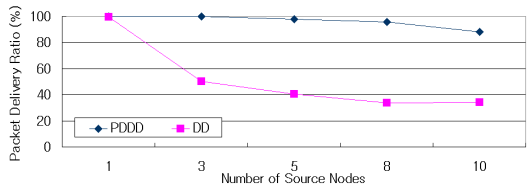
Figure 9. Performances vs. mobility of sink nodes.



(a) Average dissipated energy per a received packet per a node



(b) Average latency



(c) Packet delivery ratio

Figure 10. Performances vs. number of source nodes per a sink node.

fore, average dissipated energy per a node is decreased while it is increased in directed diffusion.

Figure 10(b) shows the average latency of successfully received data packets at the sink node versus the number of source nodes per a sink node. As the number of source nodes increased, latency of directed diffusion is rapidly increased while remaining low in PDDD. However, latency of PDDD is also rapidly increased if the number of source nodes per a sink node is 10 because packet collision is frequently occurred on paths from sources to the sink node. Although latency is rapidly increased in PDDD when the number of source nodes per a sink node is 10, latency of PDDD is still much less than latency of directed diffusion.

Figure 10(c) shows the average packet delivery ratio.As the number of sensor nodes is increased, the packet delivery ratio of directed diffusion is rapidly decreased while PDDD delivers almost all packets to the sink nodes. Note that decrement of packet delivery ratio when the number of source nodes per a destination is about 10 is mainly caused by contention of channel access.

5 Conclusion

In this paper, we propose a new data dissemination algorithm, referred to pseudo-distance data dissemination(PDDD). PDDD supports mobile sink nodes efficiently (by using pseudo-distance concepts) while previous methods such as directed diffusion do not. The simulation results conducted show that the performance of PDDD is superior to directed diffusion. For future work, we are investigating the possibility of modifying PDDD to disseminate data packets using nodes with higher remaining battery life in order to extend the nework lifetime of a wireless sensor network.

References

[1] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, Vol.11, No. 1, pp.2–16, 2003.

[2] F. Ye, H. Luo, J. Cheng, S. Lu and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *Proceeding of ACM/IEEE MOBICOM'02*, pp.148–159, Sep. 2002.

[3] M.-G Lee, S. Lee, "A pseudo-distance routing algorithm for mobile ad-hoc networks," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E89-A No.6 pp.1647–1656, Jun. 2006.