Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Communications xx (2005) 1–15
computer communications
www.elsevier.com/locate/comcom

# Energy-efficient differentiated directed diffusion (EDDD) in wireless sensor networks

Min Chen, Taekyoung Kwon*, Yanghee Choi

*School of Computer Science and Engineering, Seoul National University, Seoul, 151-744, South Korea*

Received 24 May 2005; accepted 24 May 2005

**Abstract**

A number of routing protocols [1] have been proposed for wireless sensor networks in recent years. Considering energy-efficiency as the primary objective, most of routing protocols focus on reducing the number of packet transmissions by clustering, leveraging geographical information, and so on. These routing protocols in sensor networks have the limitation of lacking application contexts for filtering or aggregation. To remedy this, Directed Diffusion (DD) [2], which utilizes application contexts in data dissemination, is proposed. However, DD cannot support time-sensitive traffic nor perform energy-balancing to increase network lifetime. To bridge this gap, this paper extends DD as follows: (1) real-time (RT) filters to provide better end-to-end (ETE) delay performance for real-time traffic, (2) best-effort (BE) filters to achieve global energy balance and to prolong network lifetime, (3) RT-repairs to fast recover node/link failure for RT traffic. The extended DD is dubbed energy-efficient differentiated directed diffusion (EDDD). Comprehensive simulation experiments show that EDDD has the following advantages: (1) differentiates dissemination service for RT and BE traffic, (2) achieves lower delay for RT traffic than DD, (3) exhibits substantially longer network lifetime than DD.


*Keywords:* Wireless sensor networks; Directed diffusion; Differentiated service; Energy-balancing

## 1. Introduction

Recent years have witnessed a growing interest in deploying a sheer number of micro-sensors that collaborate in a distributed manner on data gathering and processing [12–17]. In contrast with IP-based communication networks based on global addresses and routing metrics of hop counts, sensor nodes normally lack global addresses. Also, as being unattended after deployment, they are constrained in energy supply (e.g. small battery capacity).

These characteristics of sensor networks require energy-awareness at most layers of protocol stacks, especially at the network layer, and make energy-efficient routing one of the technical challenges. To address such challenges, most of researches focus on prolonging the network lifetime, allowing scalability for a large number of sensor nodes, or supporting fault-tolerance (e.g. sensor's failure and battery depletion) [18–23].

Some applications in sensor networks have the requirement of timely delivery. For example, when a target enters an area of interest, the delay to report the sensed event could be critical. If the reported event is not received by the sink node within the deadline, the end-to-end (ETE) delay requirement is not satisfied. After locating and detecting the target, sensor nodes may periodically report that event to a sink node. Reducing delay is more important than reliable transmission depending on the requirement of real-time (RT) applications.

Though many mechanisms have been proposed for routing delay-sensitive data in IP-based and ad hoc networks, they cannot be directly applied to wireless sensor networks. Currently, only little researches have been done on QoS routing in wireless sensor networks. In [4,5], K. Akkaya et al. investigate the additional challenges posed by imaging sensors. In [4], they use a Weighted Fair Queuing (WFQ) based packet scheduling technique to achieve the

* Corresponding author.
*E-mail addresses:* mchen@mmlab.snu.ac.kr (M. Chen), tkkwon@snu.ac.kr (T. Kwon), yhchoi@snu.ac.kr (Y. Choi).

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2005.05.019

ETE delay bound, and further consider energy efficiency and sink mobility. In [5], they proposed an energy-aware QoS routing protocol for sensor networks. It finds a least-cost, delay-constrained path for real-time data in terms of link cost that captures nodes' energy reserve, transmission energy, error rate and other communication parameters. In [6], the Sequential Assignment Routing (SAR) protocol creates multiple trees, each of which makes one-hop neighbor of the sink as root and is formed by taking into consideration QoS metric, energy resource on each path and priority level of each packet. In [7], SPEED is an adaptive real-time routing protocol that aims to reduce the end-to-end deadline miss ratio in sensor networks.

However, most of information dissemination proposals are focused on routing at layer 3; therefore, they do not have enough application contexts for filtering or in-network processing (e.g. routing assisted by application-specific code, in-network aggregation, data fusion, collaborative signal information processing).

To remedy this, [2] proposed Directed Diffusion (DD), which presents a new paradigm based on a holistic approach. In DD, the publish/subscribe mechanism provides an application's view to a sensor network, and attributed-based naming specifies which sources and sinks communicate and how intermediate nodes perform in-network processing. Sinks send interest messages to find sources; then, sources use exploratory data messages to reply to sinks. Sinks use positive and negative reinforcement messages to select or prune the path. Filters allow application-specific code to run in the network and assist diffusion and processing.

DD can be extended in many aspects, for example, geographic information in GPSR [8], GEAR [9], reliability support in RMST [10], object tracking by means of information-driven tracking filter [11]. However, DD still has some missing points such as QoS provisioning, global energy balancing and fast failure recovery.

To bridge this gap, this paper extends DD as follows: (1) real-time (RT) filters to provide better ETE delay performance for real-time traffic, (2) best-effort (BE) filters to achieve global energy balance and to prolong network lifetime, (3) RT-repairs to reduce time to fix node/link failure for RT traffic, and (4) BE-repairs to fix failure by flooding failure notification to set up new route again. The proposed DD extended by the above mechanisms is dubbed energy-efficient differentiated directed diffusion (EDDD), which has the following key properties: (1) when both RT and BE traffic coexist, the proposed two filters provide differentiated data dissemination service (normally the paths for RT and BE traffic are different), (2) EDDD achieves lower delay for RT traffic than DD and hence meets requirements for RT traffic in terms of the maximum hop count, (3) EDDD exhibits substantially longer network lifetime than DD.

The rest of this paper is organized as follows. Section 2 presents the original directed diffusion scheme. We describe RT-filters and BE-filters of EDDD in Section 3. Section 4 proposes two new repair mechanisms of EDDD. Simulation results are explained in Section 5. Finally, Section 6 will conclude the paper.

## 2. Directed diffusion (DD) overview

Directed diffusion [2,3] is a data centric dissemination protocol for sensor networks. It provides mechanisms: (a) for a sink node to flood a query toward the sensors of interest (say, sensors detecting event), (b) for intermediate nodes to set up gradients to send data along the routes toward the sink node. Diffusion provides high quality (e.g. lowest latency) paths, but requires an initial flood of the query over the entire network to explore paths. In DD, this publish/subscribe mechanism provides an application's view to all the nodes in the sensor network, and attribute-based naming specifies which sources and sinks communicate and how intermediate nodes perform in-network processing (say, filtering).

Attributes describe the data that a sink node desires by specifying sensor types, desired data rate, geographical region and so on. A monitoring node becomes a sink, creating attributes of an interest packet specifying a particular kind of data. The interest packet is then propagated over the network towards target sensor nodes (say, in the specified region). A key feature of directed diffusion is that every sensor node can be application-aware, which means that nodes store and interpret interest packets, rather than merely forwarding them along the route. Each sensor node that receives an interest packet maintains a table that contains which neighbor(s) sent that interest. To such a neighbor, it sets up a gradient. A gradient is used to evaluate the eligibility of a neighbor node as a next-hop node for data dissemination. After setting up a gradient, the sensor node redistributes the interest packet by broadcasting.

As interest packets travel across the network, sensors that match interests are triggered and the application activates its local sensors to collect and send data. In [3], original DD is extended to a DD protocol family, which includes: (1) two-phase pull diffusion, (2) push diffusion, (3) one-phase pull diffusion, each of which are summarized hereafter.

### 2.1. Two-phase pull diffusion

Two-phase pull diffusion is the original DD. In this framework, a sink sends interest messages to find sources. Sources reply with exploratory data messages to maintain paths toward the sink, and the sink uses positive and negative reinforcement messages to select or prune parts of the path. On receipt of the initial data message from the source, each intermediate node marks the message as exploratory and forwards it to all neighbors to which

gradients are set up. The initial flooding of the interest, together with the flooding the exploratory data, constitutes the first phase of two-phase pull diffusion.

If the sink has multiple pervious hop nodes, it chooses a preferred neighbor to receive subsequent data messages for the same interest (for example, the one which delivers the data message earliest). In so doing, the sink reinforces the preferred neighbor, which, in turn reinforces its preferred previous-hop node, and so on. The sink may also negatively reinforce its current preferred previous-hop node if another previous-hop node delivers the sensor data earlier. The path reinforcement and the subsequent transmission of data along the reinforced path constitute the second phase of two-phase pull diffusion.

### 2.2. Push diffusion

In one-phase push diffusion, the roles of the source and sink are reversed. Sinks become passive with interest information kept locally (e.g. to which sensory data the sink subscribe), while sources take initiatives. Exploratory data is sent over the network without interest-created gradients. A benefit of push diffusion compared to two-phase pull is that it has only one phase where information is sent throughout the network (exploratory data) rather than two phases (interests and exploratory data). Push is optimized for a different class of applications (e.g. applications with many sources and sinks), but where sources produce data only occasionally. Push is not a good match for applications with many sources continuously or frequently generating data since such data dissemination could be unnecessary.

### 2.3. One-phase pull diffusion

In one-phase pull, when an interest arrives at a source, it does not mark its first data message as exploratory, but instead sends data only on the preferred gradient. The preferred gradient is determined by the neighbor who first sends the matching interest, thus suggesting the lowest latency path. Thus one-phase pull does not require reinforcement messages, and the lowest latency path is implicitly reinforced. Notice that one-phase pull assumes symmetric communication link between nodes, which is often not true in unstable wireless networks.

## 3. RT-filter and BE-filter in EDDD

Two-phase pull DD needs both interest and exploratory data flooding, which incurs substantial ETE delay, and node/link failure recovery is not fast enough to support time-sensitive traffic transmissions. Meanwhile, one-phase push DD is designed only for some special applications where data packets are always flooded, so that they consume much more energy though the ETE delay can be very low. Among these three schemes, one-phase pull DD achieves a compromise of delay and energy consumption, which is the basis for our EDDD.

In this section, we describe RT-filters and BE-filters to extend DD. Using two kinds of filters, EDDD can differentiate data dissemination of RT and BE traffic and prolong network lifetime. By comparison, in traditional DD, data path is determined by lowest-latency (lowest ETE delay) interest or exploratory data messages, across the network. In the paper, we call the lowest-latency filter the traditional DD filter.

### 3.1. RT-gradients and BE-gradients Overview

RT and BE filters are realized by setting up corresponding gradients, which is performed when receiving interest packets. The information contained in an interest packet is shown in Fig. 1. The description of each packet field is shown in Fig. 3.

Fixed attributes in Fig. 1 specify which sources and sinks communicate. Whenever a sink initiates the new interest flooding periodically, it will increment its counter, ISeqNum. The fixed attributes are not changed while propagated across the network. On the other hand, when an intermediate node broadcasts an interest packet, it will change variable attributes. The TTL field means how many hops the interest packet can be propagated more. In particular, it is used for RT traffic to limit the overall path length, and must be initialized by the sink considering the delay requirement of RT traffic since larger TTL permits larger ETE delay between a source-sink pair. RT-gradient_UpdateFlag (RUF) and BE-gradient_UpdateFlag (BUF) in Fig. 1 are used to indicate which kind of gradients (RT or BE) needs to be updated.

| INTEREST Packet |
|---|
| ***Fixed Attributes*** |
| SinkID<br>Application Context (e.g. key, type, operator, value)<br>ISeqNum<br>Flow-id |
| ***Variable Attributes*** |
| TTL (*only for RT traffic*)<br>PreviousHopID<br>PreviousHopEnergy |
| ***Neighbor Information Table Update Flag*** |
| RT-gradient_UpdateFlag(RUF)<br>BE-gradient_UpdateFlag(BUF) |
| ***Gradients*** |
| HopCount(HC)<br>PreviousMPE |

Fig. 1. Interest Packet Format.

Gradients attributes are used to setup/update gradients for either RT-filter or BE-filter. To solve the energy balancing issue, we introduce a new kind of gradient: *Minimum-Path-Energy* (MPE). The *MPE* gradient of a path (toward the sink) is the minimum energy level of the nodes along the path. It is mainly designed for BE-filters to achieve load balancing, while the *HopCount* (HC) gradient is introduced to choose the shortest path to provide lower delay for RT traffic. Here, *HC* is the number of hops from the sink to an intermediate node. However, both gradients are considered when these two filters are set up. RT-filters initially find a list of candidate paths whose *HC* is minimal. Among these paths, the RT-filter will choose the path (toward the sink node) whose MPE is the maximum. BE-filters deal with these two gradients with reversed priority. These two gradients cooperate to balance the tradeoff between energy efficiency and lower delay so as to differentiate RT and BE traffic.

Note that, HC and MPE are 'per-path gradients,' which means that information about the path is considered and updated when the interest packets are flooded over the network. By comparison, traditional gradients in original DD [2] can be deemed as 'per-node gradients'.

### 3.2. BE-gradients setup

Fig. 3 explains how an intermediate node handles an arriving interest packet for BE traffic. When an intermediate node receives an interest packet, first it will look at the information contained in the interest packet. *PreviousHopID* in the interest packet is the index for the corresponding *neighbor information entry* (NIE) in Fig. 2. The collection of NIEs is called *neighbor information table* (NIT). The ID is the unique identification of the neighbor (e.g. IEEE MAC address).

The intermediate node will update its NIE depending on which kind of update is needed. *BE-gradient_UpdateFlag* (BUF) is a flag that indicates BE-gradient update is needed, while *RT-gradient_UpdateFlag* (RUF) is a flag that indicates RT-gradient update is needed. One of the two flags is set in any interest packet. An interest packet with *BUF/RUF* set is used to setup/update BE-gradients/RT-gradients, respectively.

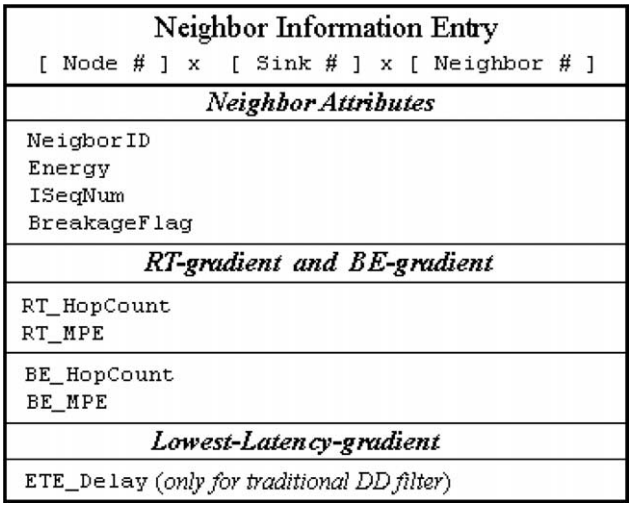


Fig. 2. Neighbor Information Entry.

The following step 4 in Fig. 3 is most important. In Fig. 4, let Path-A consist of nodes ***s***, ***o***, and ***p***; Path-B consists of nodes ***s***, and ***m***; Path-C consists of nodes ***s***, ***u***, and ***v***. Then Path-A+, Path-B+, Path-C+ are the resulting path made by the union of node ***i*** (including the link between node ***i*** and the previous-hop node) and Path-A, Path-B, Path-C, respectively. When the intermediate node ***i*** receives an interest packet from the previous-hop node ***p***, it will calculate its *CurrentMPE* (MPE of Path-A+, which is equal to minimum[energy-level(node i), MPE of Path-A]). If this value is the largest at the moment (for example, it is larger than those of Path-B+ and Path-C+), node ***i*** considers that node ***p*** (in Path-A+) is the best node to set up a BE-gradient. Thus, it will set *BUF* in the interest packet and rebroadcast the packet in order to let its neighbors update their BE-gradients to node ***i***, if necessary.

In Fig. 5(a), a BE flow entry stores the best BE-gradient to look up the next hop node fast. In our design, multiple sink nodes are supported. Each sink node can also generate multiple flows (e.g. RT flows and BE flows). A sink node periodically initiates interest-flooding to pull a data flow with a new interest sequence number. *Interest sequence number* (ISeqNum) is incremented by 1 whenever the sink initiates flooding of an interest packet. When an intermediate node receives an interest packet with larger *ISeqNum*, it will update the corresponding NIE and flow entry or setup a new one.

If node ***i*** receives the first interest packet in Fig. 5(b), it operates as follows: (1) sets up a BE-gradient to node ***p***; (2) updates its BE flow entry, and chooses node ***p*** as its next hop node; (3) sets BUF in the interest packet. Note that an intermediate node will wait for a random small time before re-broadcasting an interest packet with *BUF* set since another interest packet with larger *MPE* could arrive soon. It also reduces the probability of collision in flooding interest packets. Then node ***i*** rebroadcasts the updated interest packet.

When it receives the second interest packet in Fig. 5(c), it operates as follows: (1) sets up a BE-gradient to node ***m***; (2) discards the interest packet since Path-B+ has less *MPE*.

When it receives the third interest packet in Fig. 5(d), it operates as follows: (1) sets up a BE-gradient to node ***v***; (2) updates its BE flow entry, and chooses node ***v*** as its next hop node (the previous next hop node ***p*** is replaced) since Path-C+ has larger *MPE*; (3) sets *BUF* in the interest packet. Note that if an intermediate node receives multiple interest packets with the same *MPE*, it will select the path with the smallest *HC*.

**Intermediate Node Handles Interest Packet with BUF Set**

**Step 1**: Get Infromation from Interest Packet

***SinkID***: the identifer of the sink node
***ISeqNum***: interest Sequence Number
***TTL***: time to live, only used by RT traffic
***PreviousHopID***: the identifer of the previous hop node
***RUF:*** RT-gradient_UpdateFlag
***BUF:*** BE-gradient_UpdateFlag
***HopCount***: hop count of the path from sink to this node.
***PreviousMPE***: the Minimum-Path-Energy of the path from sink to previous hop node

**Step 2 : Find NIE in NIT according to PreviousHopID**

***NIT***: Neigbhor Information Table
***NIE***: Neighbor Information Entry

**Step 3: In the NIE, update gradient for best-effort traffic**

**Step 4 : Decide whether the interest should be broadcast or not , and update the *BE Flow Entry***

***CurrentMPE:*** the Min-Path-Energy of the path from sink to this node. The path traverses the previous-hop node.
***MaxMPE***: maximum MPE among all the previous-hop nodes. It's updated based on CurrentMPE.
***HC_with_MaxMPE***: hop count of the path with MaxMPE.
/*Note: *MaxMPE* and *HC_with_MaxMPE* are best gradients for BE traffic in the history of arriving interest packets */

```
CurrentMPE = min(Self_Energy, PreviousMPE);
Case 1:  "First time to receive the interest with new SinkID or Flow-id or ISeqNum":
    MaxMPE = CurrentMPE;   HC_with_MaxMPE = HopCount;  NextHopNode = this node
        /* Update BE flow entry*/
Case BE-1: "CurrentMPE > MaxMPE"
    MaxMPE = CurrentMPE;   HC_with_MaxMPE = HopCount;  NextHopNode = this node
        /* Update BE flow entry*/
Case BE-2: "(CurrentMPE == MaxMPE)&&(HopCount < HC_with_MaxMPE)
    HC_with_MaxMPE = HopCount;  NextHopNode = this node
        /* Update BE flow entry*/
```

**Step 5**: If one of the above three cases happens, update the interest packet and broadcast it.

Fig. 3. Pseudo-code for BE-gradients Setup.

### 3.3. RT-gradients setup

Fig. 6 shows the operation to setup a RT-gradient. Step 1, 2, 3 are the same as the ones in Section 3.2; only Step 4 is different. Fig. 7 is an illustration. Fig. 7(a) indicates the information contained in a RT flow entry. If node ***i*** receives the first interest packet as depicted in Fig. 7(b), it operates as follows: (1) sets up a RT-gradient to node ***p***; (2) updates its RT flow entry, and chooses node ***p*** as its primary next hop node; (3) sets *RUF* in the interest packet. In the case of RT-gradients, the first interest packet is broadcast as soon as possible, since timely delivery is the key concern.

When it receives the second interest packet in Fig. 7(c), it operates as follows: (1) sets up a RT-gradient to node ***v***, (2) updates its RT flow entry, and chooses node ***m*** as its primary next hop node (then node ***p*** becomes a backup next hop node in case of primary node failure) since Path-B+ has less *HC*; (3) sets *RUF* in the interest packet. Note that an intermediate node maintains not only primary next hop node, but also backup next hop node for fast failure recovery for RT traffic (in Section 4).

When it receives the third interest packet as depicted in Fig. 7(d), it operates as follows: (1) sets up RT-gradient to node ***v***, (2) discards the interest packet since Path-C+ has larger *HC* than Path-B+, (3) chooses node ***v*** as its another backup next hop node. Note that if an intermediate node receives multiple interest packets with the same *HC*, it will select the path with the largest *MPE*.

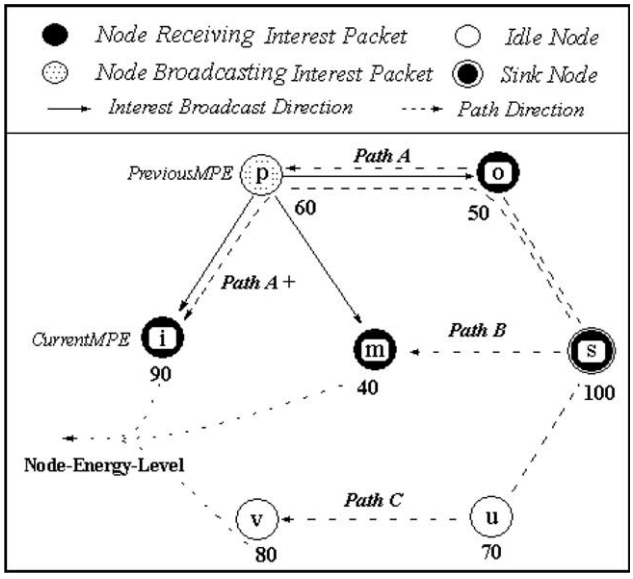


Fig. 4. Intermediate nodes calculate *CurrentMPE* when receiving an interest packet.

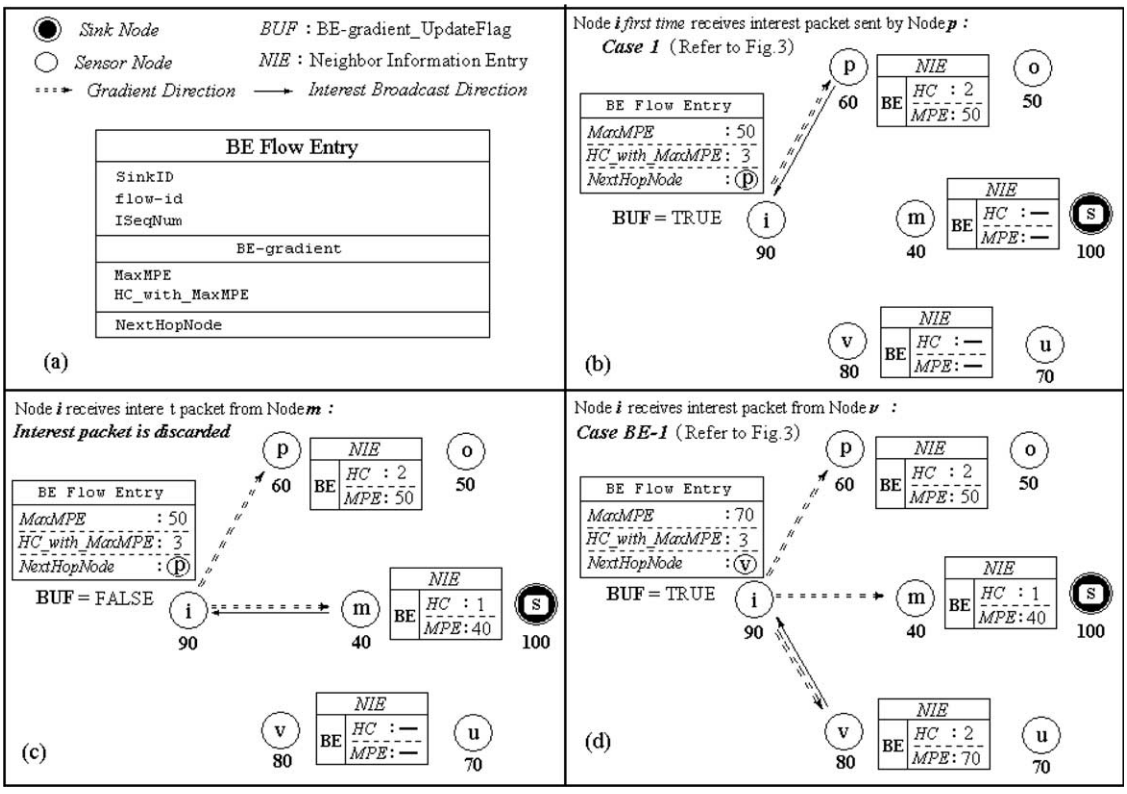


Fig. 5. Intermediate Nodes setup a BE-gradient and update a BE-flow-entry when receiving interest packet with BUF set.

In Figs. 3 and 6, Case 1 is the traditional case of directed diffusion; Case BE-1 and RT-1 indicate *CurrentMPE* and *HC* are largest at that moment, respectively; Case BE-2 and RT-2 indicate one gradient with higher priority is equal to the previous largest value, then the other gradient needs to be compared. We refer to this scheme as a multiple-level gradient mechanism. RT-filters consider the HC gradient with the higher priority, while BE-filters consider the *MPE* gradient prior to *HC* gradient. This two-level priority is realized by introducing *MaxMPE* and *HC_with_MaxMPE* (in Fig. 3) for BE traffic, while *MinHC* and *MPE_with_-MinHC* (in Fig. 6) for RT traffic.

Compared to setting up 'per-node gradient' in original DD, we add four more cases to re-broadcast interest packets in order to setup/update 'per-path gradient' in terms of energy-balancing and/or hop-count, which requires more

**Intermediate Node Handles Interest Packet with RUF Set**

**Step 1**: Get Infromation from Interest Packet

**Step 2** : Find NIE in NIT according to PreviousHopID

**Step 3** : In the NIE, update gradient for real-time traffic

**Step 4** : Decide whether the interest should be broadcast or not , and update the *BE Flow Entry*

***MinHC***: minimum hop count among all the previous-hop nodes.
***MPE_with_MinHC***: CurrentMPE of the path with MinHC
/*Note: *MinHC* and *MPE_with_MinHC* are best gradients for RT traffic in the history of arriving interest packets */

```
CurrentMPE = min(Self_Energy, PreviousMPE);
Case 1:   "First time to receive the interest with new SinkID or Flow-id or ISeqNum":
    MinHC = HopCount;      MPE_with_MinHC = CurrentMPE;
        /* Update RT flow entry*/
Case RT-1 : "HopCount < MinHC":
    MinHC = HopCount;      MPE_with_MinHC = CurrentMPE;
        /* Update RT flow entry*/
Case RT-2 :   "(HopCount == MinHC)&&(CurrentMPE > CurrentMPE_with_MinHC)":
    MPE_with_MinHC = CurrentMPE;
        /* Update RT flow entry*/
```

**Step 5**: If one of the above three cases happens, update the interest packet and broadcast it.

Fig. 6. Pseudo-code for RT-Gradients Setup.

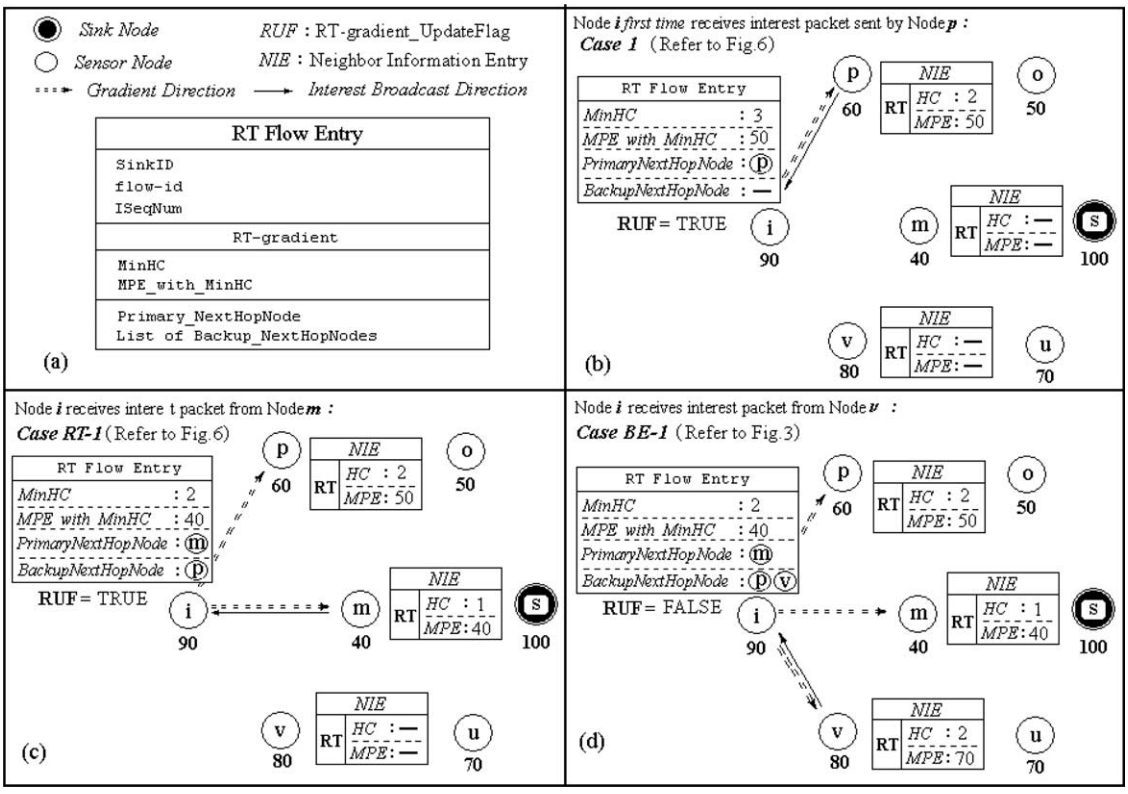


Fig. 7. Intermediate Nodes Setup RT-gradients and update RT-flow-entry when receiving interest packets with RUF set.

control messages to be setup entirely. However, this cost is relatively small considering the fact that gradient setup is a much less frequent task compared to the frequent data dissemination interval.

### 3.4. RT and BE-filters to differentiate data dissemination

An arriving data packet will trigger an intermediate node to look at the attributes in the packet. Whether a RT-filter or a BE-filter is chosen depends on the QoS attribute (in Fig. 8). Then the specified filter will access the corresponding flow entry to use the best next hop in forwarding the data packet.

For BE traffic, only the node cached in the BE flow entry (in Fig. 5) will be used as the next hop node. For RT traffic, if the primary next hop node is available in the RT flow entry (in Fig. 7), it will be used as the next hop node. Otherwise, the one with *MinHC* will be chosen among the backup next hop nodes.

There can be multiple paths between a source-sink pair. Fig. 9 illustrates multiple possible paths between sources and a sink. Suppose path A-1, A-2 and A-3 are the shortest paths that constitute a *path group* (PG-A) with *MinHC*, among which the one with maximum *MPE* will be chosen for RT packets.

If all the paths in PG-A are blocked (say, energy depletion in some nodes), the second best *PG* (PG-B) will be chosen to disseminate RT traffic, and so on. Shortly speaking, RT-filter seeks to lower delay first, then seeks to achieve *local energy balance* within a *PG*. We can illustrate the sequence of the used paths to transmit RT traffic is: A-1, A-2, A-3, A-1, A-2, A-3,…(energy depletion in PG-A), B-1, B-2, B-1, B-2,…(energy depletion in PG-B), C-1, C-2, C-1, C-2, and so on. Note these paths (or path groups) are not strictly disjoint.

Since delay is not considered with higher priority for BE traffic, BE-filter always chooses the path with maximum *MPE* on which the minimum energy level is the maximum and seeks to achieve *balanced energy consumption* over all sensor nodes. An illustration for the sequence of the used paths to transmit BE traffic could be: A-1, A-2, A-3, B-1, B-2, C-1, C-2,…, E-1, E-2, A-1, A-2,… Note that the above illustration assumes that all the sensor nodes have the same battery life in the beginning.

We carry out numerical experiments using EDDD by the simulation model as detailed in Section 5. Fig. 10 shows that

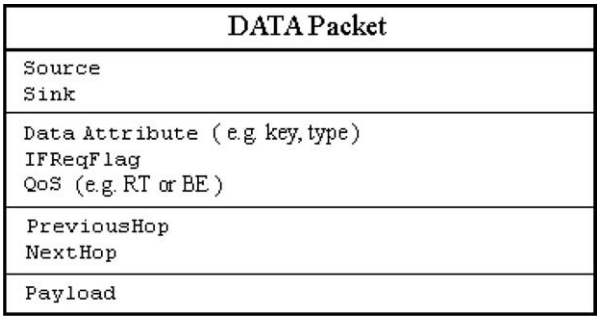


Fig. 8. Data Packet Format.

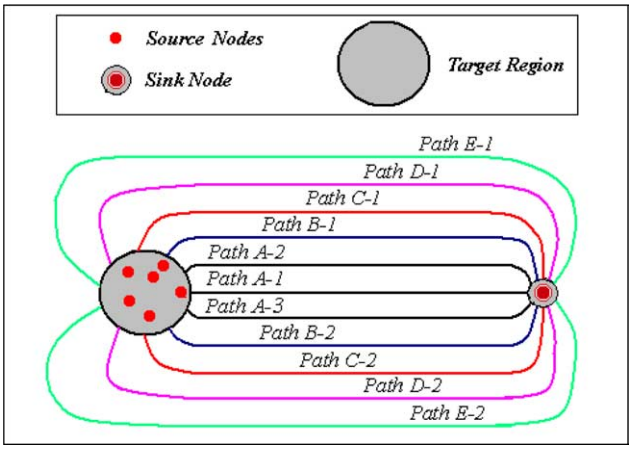


Fig. 9. Packet Groups Split by RT and BE-filters.

ETE packet delay of RT traffic is step-wise increasing. The different *delay levels* (DL) correspond to different PGs that have different hop counts. In the same *DL*, the delay fluctuation is caused by flooding interest packets to choose a better path within the same PG, which aims at local energy balance. If the delay of a certain DL is too high to meet the requirement for time-sensitive transmission, we say the sensor network cannot support such QoS requirement anymore. Recall that there is a TTL field to limit the path length between sensors and the sink node to satisfy delay requirement.

Fig. 11 shows the ETE packet delay of BE traffic. We observe that delay fluctuates more significantly than that of RT traffic since BE-filter chooses the maximum *MPE* path among all available paths between the source-sink pair. Then in the next round of interest flooding, the selected path is likely to be replaced by other paths since it has consumed some energy in the previous round while the other paths have not. Thus, BE-filter works like the *Round-Robin* path scheduling algorithm among the paths across the network.

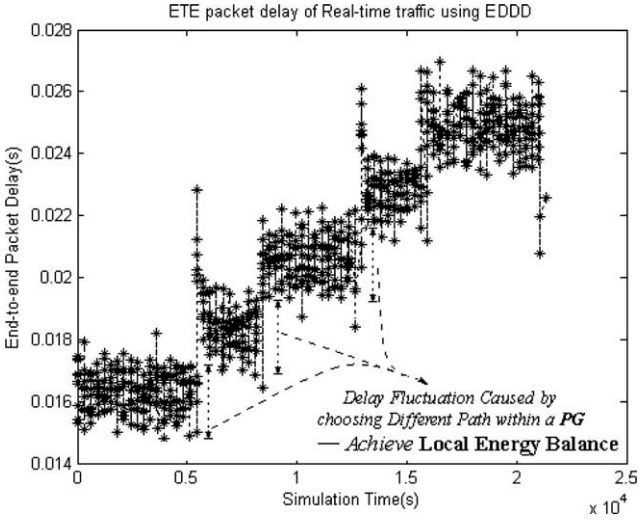


Fig. 10. ETE packet delay of RT traffic using EDDD.

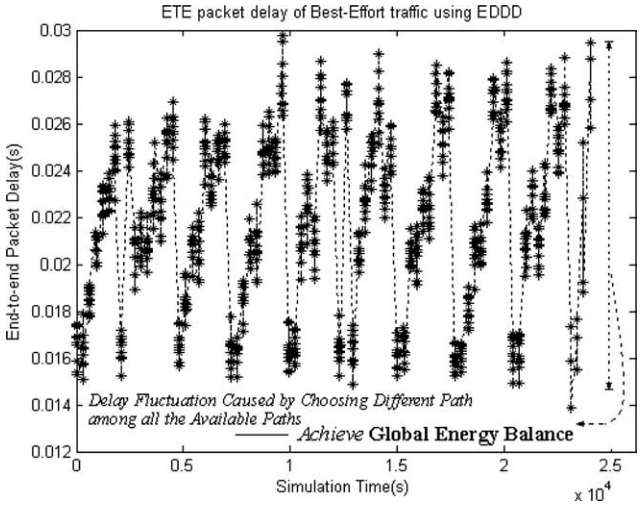


Fig. 11. ETE packet delay of BE traffic using EDDD.

## 4. RT-repair and BE-repair mechanisms in EDDD

Among the three previous DD frameworks in Section 2, there is no local repair mechanism in one-phase pull diffusion and push diffusion to handle network changes (due to node failure, energy depletion, or mobility). Even in two-phase pull diffusion, the source periodically sends additional exploratory data messages to adjust gradients. Then the sink propagates negative reinforcement to tear down the existing path and positive reinforcement to reinforce a new path, respectively. In this mechanism, link recovery is so slow that it is hard to meet the requirement for time-sensitive traffic.

In case of link failure, we propose the following local repair mechanisms: (1) for real-time traffic, the best available neighbor is chosen to achieve fast recovery, (2) for best-effort traffic, failure notification is flooded to setup new route again.

In this section, we assume that there is a retransmission mechanism based on acknowledgement packets in media access control (MAC) protocol for reliability. If a predetermined number of retransmission fails, the MAC layer informs this failure to the upper layer, which is EDDD entity.

### 4.1. BE-repair

If MAC feedback information indicates that transmitting a BE data packet to the next hop node fails, the intermediate node will mark the next hop node broken in neighbor information entry. Since ETE delay is not a primary concern, the intermediate node will flood a *BreakageNotification* packet. When the sink receives the *BreakageNotification* packet, it will initiate interest flooding immediately to update stale gradients over the network.

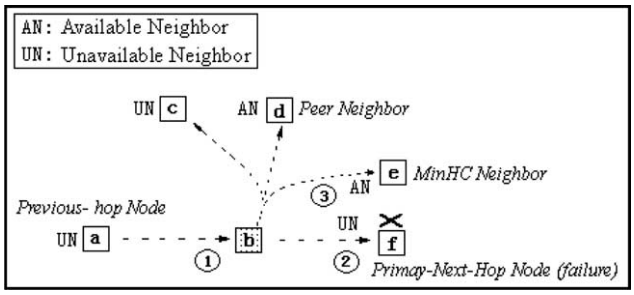


Fig. 12. Intermediate Node Chooses Second Best Neighbor For RT packet.

### 4.2. RT-repair

In Fig. 12, node ***b*** receives a RT data packet from ***a*** (***b***'s previous hop node), it will choose its primary next hop node ***f*** in the RT-flow-entry (in Fig. 7) to forward the packet. If ***f*** fails to forward the packet, ***b*** will mark ***f*** broken in *NIE*.

To provide better ETE delay performance for RT packets, ***b*** will find the best neighbor e among all the *available neighbors* (ANs). ANs are the nodes that can ensure no loop if the intermediate node (***b*** in Fig. 12) chooses one of them as the next hop node. By comparison, *unavailable neighbors* (UNs) are the nodes that do not belong to ANs. The choice of ANs is described in Fig. 13. We deem the neighbor whose hop count is equal to MinHC + 1 as the peer neighbor.

Interest flooding is triggered to update gradients when the current route becomes unstable due to the battery exhaustion of some nodes on the data path. In Fig. 13, if there is only one AN left, the intermediate node will set *IFReqFlag* (Interest Flooding Request Flag) in the RT data packet.

When the sink receives an RT packet containing *IFReqFlag*, it will initiate interest flooding immediately to update stale gradients. When all the neighbors of an intermediate node are UNs, it drops the data packet, and floods a *BreakageNotification* packet (same as BE-repair). When all the neighbors of the source node are broken, we say that moment is network lifetime since there are no available paths between source-sink pair anymore. We call such lifetime *LifeTimeTypeII*. By comparison, we call the time that the first node dies *LifeTimeTypeI*. Our simulation will compare these two kinds of lifetime both in original DD and EDDD.

## 5. Numerical results

### 5.1. The simulation model

Fig. 14 shows our sensor network model. Three hundreds of sensor nodes are randomly distributed on a 200 m × 200 m area. The sensor nodes are battery-operated except the sink node. The maximum transmission range of sensor node is 15 m. We assume both the sink and sensor nodes as stationary. A sink node is assumed to have infinite energy supply. It is located close to one corner of the area, and the task region is specified at the other corner.

The sink node will initiate interest flooding (indicates a new task) periodically. Interest is propagated on a hop-by-hop basis towards sensor nodes in the target region, which is depicted by a gray circle in Fig. 14. At any moment, one of the sensor nodes within the target region matches the interest and sends data to the sink node.

Fig. 15 shows the protocol stack of our sensor node model; it includes application layer, routing layer, data link layer and physical layer. In the application layer, a sensor can generate both real-time and best-effort traffic. Each task requires periodic transmission of data packets with a *constant bit rate* (CBR) of 100 packet/sec. There are two kinds of flows generated based on the ratio of RT traffic to

**RT-Repair Mechanism**

***MinHC***: Minimum Hop Count ***Peer Node***: the node whose hop-count is equal to MinHC +1
***PeerTransmissionFlag***: set in the data packet when an intermediate node forward it to its peer node
***AN***: Available Neighbor, includes: (1) MinHC neighbor, (2) peer neighbor
***UN***: Those who do not belong to ANs are deemed as unavailable Neighbors
***#UN***: Number of UN ***#AN***: Number of AN
***IFReqFlag***: Interest Flooding Request Flag

*If the information feedback from MAC layer indicates the next hop node fails to transmit RT data packet*
**Step 1:** *Set it as breakage neighbor;*
**Step 2:** */*Determines possible ANs to ensure no loop*/*
*If (PeerTransmissionFlag == 1), ANs* = {MinHC neighbors}
*/* data packet is sent by its peer node*/*
*Else ANs* = {MinHC neighbors} ⋃ {peer neighbors}
**Step 3:** *If (#AN == 0)*
*Drop RT data packet, flood BreakageNotification Packet immediately.*
*Else*
*Find the best next hop node among ANs;*
*If (#AN == 1) Set IFReqFlag in the RT data packet header;*
*If (the next hop is a peer node) Set PeerTransmissionFlag in RT data packet header;*
*Retransmit the lost data packet.*

Fig. 13. Pseudo-code for RT-Repair Mechanism.

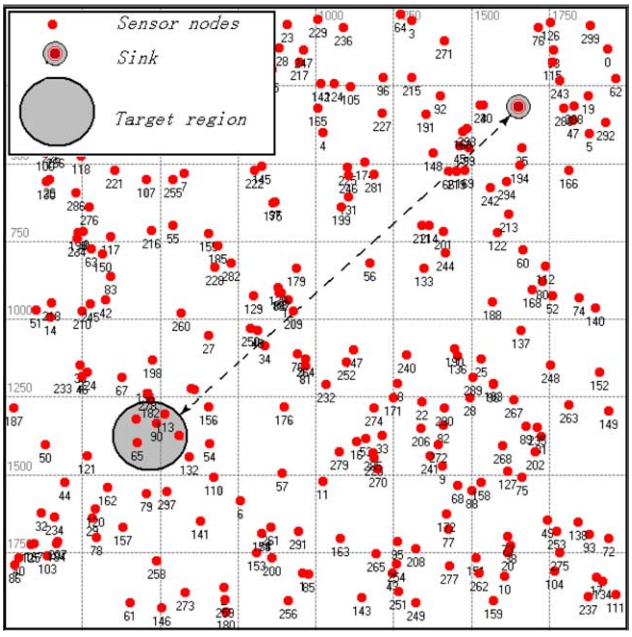


Fig. 14. Sensor Network.

BE traffic (or *traffic mix ratio*, TMR). RT&BE-filters and local repair mechanisms are implemented at the routing layer. Energy-efficient MAC protocol (e.g. S-MAC [25]) at 2 Mbps is used as the MAC layer protocols.

The parameters we used in our simulation are shown in Tables 1 and 2. We deploy an energy model according to the power consumption parameters in [26]. Every node starts with the same initial energy budget (4,500 W sec). We can use the following equation to calculate the energy consumption in a state (transmitting, receiving, or over-hearing): $m \times \text{PacketSize}_{\text{MAC}} + b + P_{\text{idle}} \times t \times 1000$ (μW sec). Note that $P_{\text{idle}}$ is in *mW* unit and hence 1000 is multiplied. In the above equation, $m$ represents the incremental cost compared to the power consumption in idle state, $b$ represents fixed cost, and $t$ represents the duration of the state. For example, transmitting one MAC packet consumes the following energy: $m_\alpha \times \text{PacketSize}_{\text{MAC}} + b_{\text{tx}} + P_{\text{idle}} \times t_{\text{tx}} \times 1000$ (μW sec). As energy of a sensor node runs out, the node will be disabled. Note that our goal is fair performance comparison between DD and EDDD. These setting are common to all the simulated schemes.

### 5.2. Performance metrics

Five important performance metrics are evaluated:

*Lifetime*, There are two kinds of lifetime, namely *LifeTimeTypeI* and *LifeTimeTypeII*. *LifeTimeTypeI* is the time when the first node dies. *LifeTimeTypeII* is the time when source nodes have no available paths to sink.

*End-to-end delay of data packets*, This includes all possible delays during data dissemination from source to sink, caused by buffering, queuing at the interface queue, retransmission delays at the MAC, and propagation and transfer times.

*End-to-end delay jitter of data packets*, The variation of ETE packet delay between successively received packets in the same flow.

*Packet delivery ratio*, The ratio of the number of data packets delivered to the sink to the number of packets generated by the source node.

*Normalized Number of Control Packets*, The ratio of number of interest and control packets transmitted to the number of data packets delivered at the sink.

Lifetime metric is the most important for sensor networks. Packet delay and delay jitter metrics are also important for time-sensitive RT traffic.

### 5.3. Performance comparison of traditional DD Filter and RT and BE-filters

For fair comparison between different filters, the simulations in this section use the same DD framework

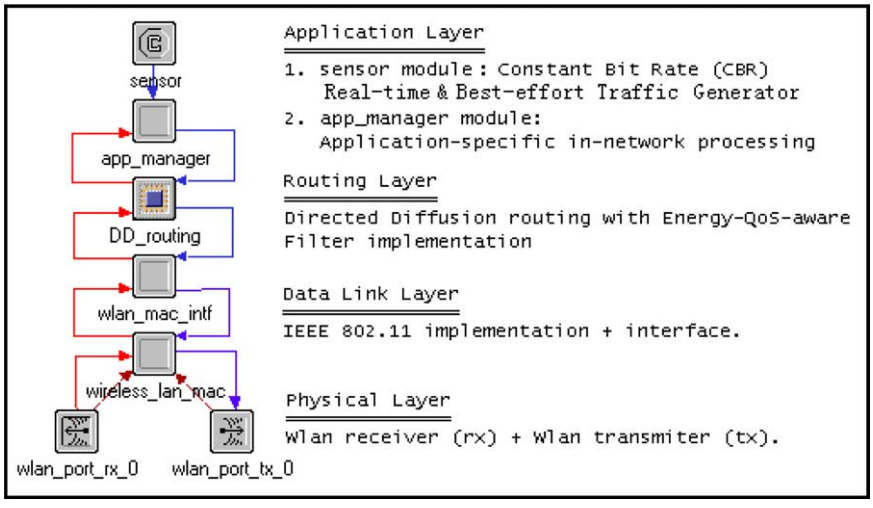


Fig. 15. Sensor Node Model.

Table 1
Simulation parameters configuration

| | |
|---|---|
| Sink periodically flooding interest interval | 300s |
| Average Sensor Packet Inter-arrival time | 10ms |
| Sensor Data Packet Payload | 1024bits |
| Data rate | 2Mbps |
| Network scale | 200m × 200m |
| Total sensor node number | 300 |
| Topology configuration mode | Randomized |
| Maximum transmission range | 15m |

Table 2
Energy consumption parameters configuration of lucent IEEE802.11 2 Mbps WaveLAN card [26]

| Normalized initial energy of sensor node | | 4500 (W .sec) |
|---|---|---|
| Incremental cost ($\mu W$ .s/byte) | $m_{tx}$ | 1.9 |
| | $m_{recv}$ | 0.5 |
| | $m_{overhearing}$ | 0.39 |
| Fixed cost ($\mu W$ .sec) | $b_{tx}$ | 454 |
| | $b_{recv}$ | 356 |
| | $b_{overhearing}$ | 140 |
| $P_{idle}$ | | 843 (mW) |

with the local repair mechanisms. Table 3 shows the comparison of the network lifetime, the average ETE packet delay, the delay jitter, the packet delivery ratio, and the normalized number of the control packets for traditional DD filter and RT and BE-filters of EDDD.

In terms of *LifeTimeTypeI*, RT-filter (Scheme B) has 1506 s more lifetime than the original DD filter (Scheme A), and BE-filter has 18044 s more lifetime than the *Scheme A*. In terms of *LifeTimeTypeII*, RT-filter has 1513 s more lifetime than *Scheme A*, and BE-filter has 4234 s more lifetime than the *Scheme A*. Note that *LifeTimeTypeI* of BE-filter is close to its *LifeTimeTypeII*.

The ETE packet delay of RT traffic is almost always lowest among all the schemes. RT-filter outperforms lowest-latency-filter. In Fig. 16, both filters provide similar delay performance in the beginning. In general, with energy consumed by continuous data forwarding, depleted sensor nodes will generate uncovered area. In original DD, lowest latency filter does not consider any energy efficiency, so that some important sensor nodes deplete itself fast and hence corresponding paths are consumed, which otherwise provide a shorter delay.

The delay performance improvement by RT-filter also verifies the usefulness of both *HC* and *MPE* gradients, which provide better ETE delay performance and achieve *local energy balance* simultaneously.

In our simulation experiments, node mobility, fluctuations in channel quality and fading/multi-path effect are not considered. *LifeTimeTypeI* is the amount of time until any node has no ability (not enough energy) to forward packets. *LifeTimeTypeII* is the amount of time until no available path exists between the source and the sink.

Note, though the total control packet number of RT and BE filters are larger than that of the traditional DD filter, the normalized number of control packet becomes smaller,

Table 3
Comparison of Performance metrics of DD and EDDD

| Abbreviation | Scheme A: Lowest latency filter (Traditional DD filter) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Scheme B: real-time filter | | | | | | | | | |
| | Scheme c: best-effort filter | | | | | | | | | |
| | Scheme D: Joint RT&BE filter (Traffic mixed ratio of RT traffic to BE traffic = 1/1) | | | | | | | | | |
| Scheme | A | | B | | C | | D | | | |
| Lifetime type | Type I | Type II | Type I | Type II | Type I | Type II | Type I | | Type II | |
| Traffic type | Not Support RT&BE | | RT | | BE | | RT | BE | RT | BE |
| Lifetime (s) | 5208 | 19734 | 6714 | 21247 | 23252 | 23968 | 11924 | | 22184 | |
| Average ETE Pk delay (ms) | 16.72 | 21.14 | 16.57 | 20.43 | 21.65 | 21.95 | 16.56 | 23.32 | 18.29 | 24.37 |
| Average ETE Pk delay jitter (ms) | 2.95 | 4.06 | 1.8 | 2.33 | 3.11 | 4.16 | 1.89 | 4.21 | 1.92 | 4.1 |
| Packet delivery ration (%) | 99.2 | 95.1 | 99.3 | 98.4 | 99.6 | 97.2 | 100 | 99.1 | 98.5 | 98.2 |
| Total control packet number | 2379 | 14879 | 11325 | 48378 | 38367 | 39209 | 28561 | | 52309 | |
| Normalized control Pk Num | 0.059 | 0.068 | 0.191 | 0.209 | 0.152 | 0.154 | 0.238 | | 0.214 | |

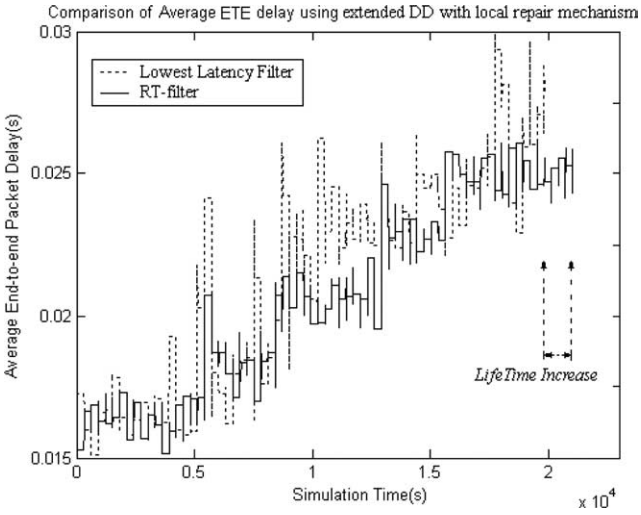


Fig. 16. Comparison of ETE packet delay between Lowest Latency Filter and RT-filter.

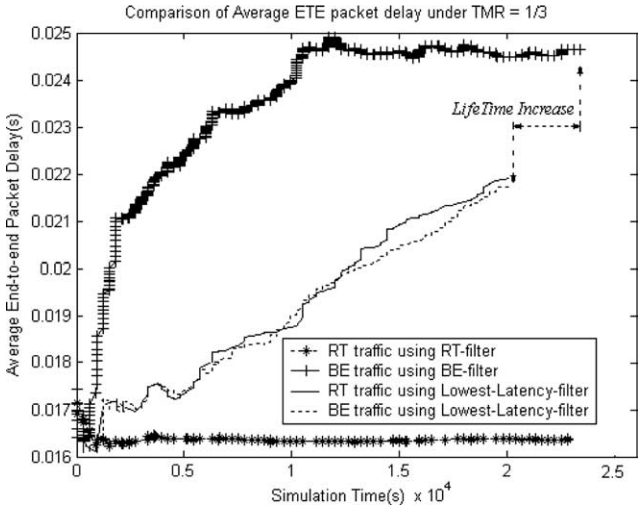


Fig. 17. Comparison of ETE packet delay using tradition DD filter and RT&BE filters under TMR = 1/3.

because interest flooding is a relatively infrequent task compared to the period of data transmission. Clearly, compared with traditional DD filters, RT and BE-filters have following improvements by increasing control overhead: (1) provide differentiated dissemination service for RT and BE traffic, (2) achieve better delay performance for RT traffic, hence further meet requirements for RT traffic, (3) exhibit longer network lifetime.

### 5.4. Performance comparison under different TMR (Traffic Mix Ratio of RT Traffic to BE Traffic) using EDDD

Since the ETE packet delay and the delay jitter are the most important for time-sensitive traffic, we choose them as the main metrics to compare between the two traffic classes under different *TMR*. First, we show RT and BE filters outperform traditional DD filters in terms of QoS provisioning in Fig. 17.

Figs. 18(a), 19(a), 20(a) show ETE packet delays under *TMR* setting to 1/1, 1/2, 1/3, respectively, while Figs. 18(b), 19(b), 20(b) show corresponding ETE packet delay jitters. In all scenarios with different *TMRs*, the sum of RT traffic and BE traffic is 1 packet/sec. We can see the following phenomena: (1) the ETE packet delay of RT and BE traffic is differentiated obviously; the ETE packet delay of RT traffic is almost always lower than that of BE traffic, (2) with *TMR* decreasing, the ETE packet delay of RT traffic also decreases, and the delay jitter becomes less fluctuating, (3) with *TMR* decreasing, the ETE packet delay jitter of BE traffic increases. The results are consistent with the analysis in Section 3.4.

While RT traffic is disseminated using some path of a *path group* (PG) with *MinHC*, energy of the chosen path is consumed. Next time when the new interest packet is flooded, BE-filters will choose some other path, since other paths have more remaining energy. So BE-filters yield PGs

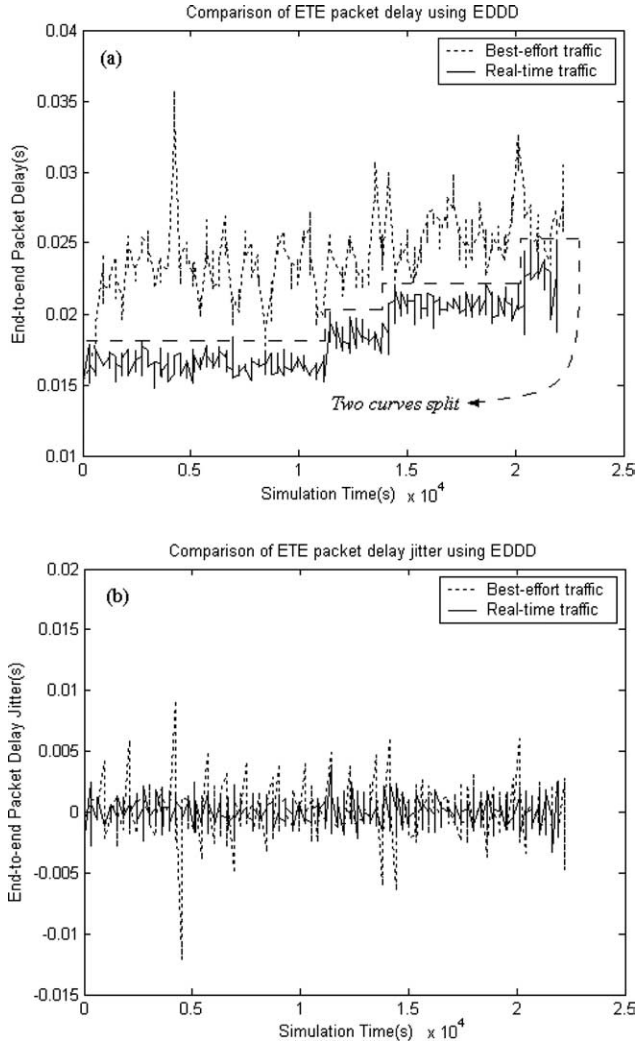


Fig. 18. Comparison of ETE packet delay and delay jitter with TMR = 1/1.

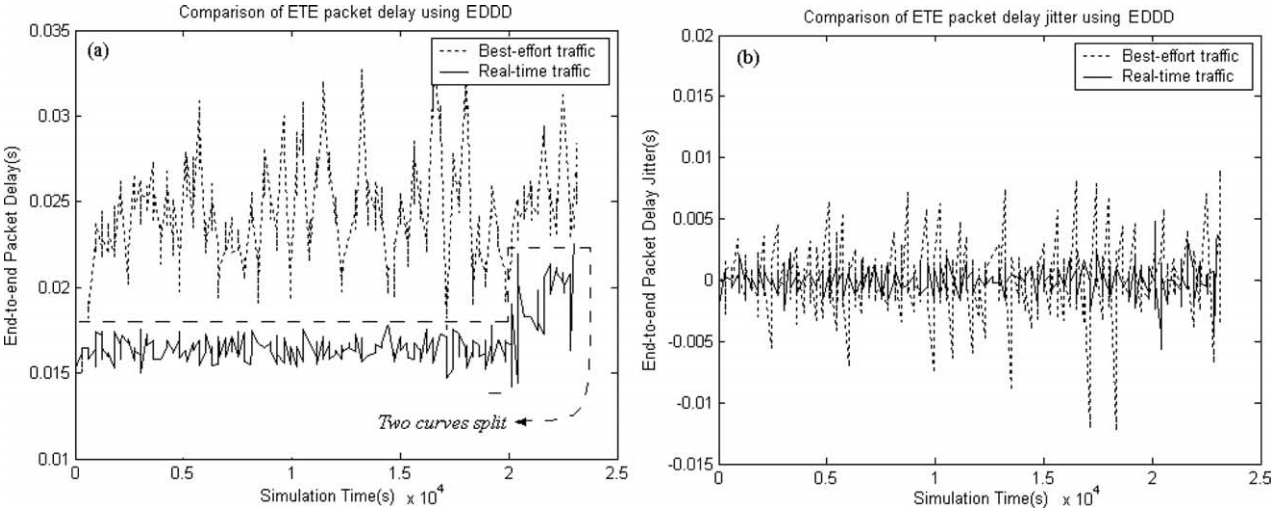


Fig. 19. Comparison of ETE packet delay and delay jitter with TMR = 1/2.

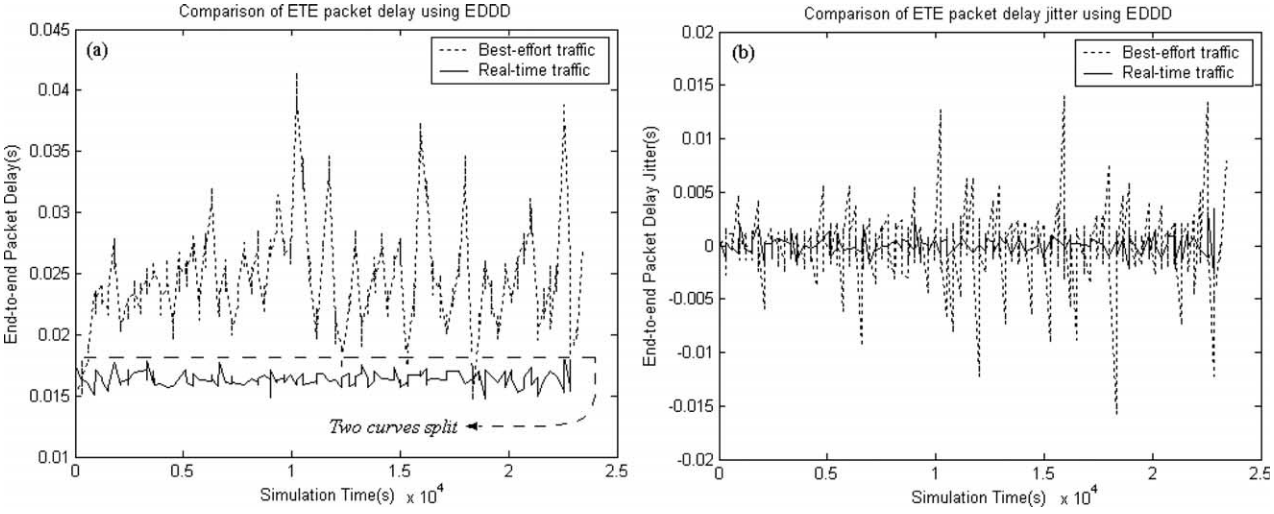


Fig. 20. Comparison of ETE packet delay and delay jitter with TMR = 1/3.

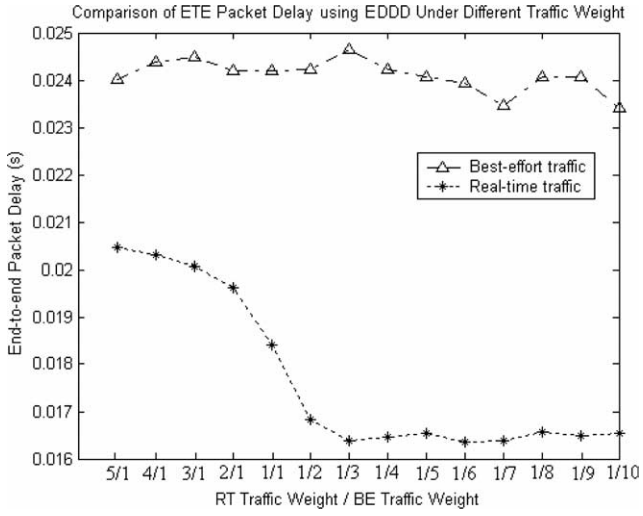


Fig. 21. Comparison average ETE packet delay using different TMR in EDDD.

with *MinHC* to RT traffic. That is the reason why the ETE packet delays of RT and BE traffic split.

When *TMR* decreases (RT traffic load becomes smaller), RT traffic will use less number of paths, leaving more available paths for BE-filter. That is why, with *TMR* decreased, the ETE packet delay of RT traffic becomes smaller and less fluctuating in Fig. 21, while the delay jitter of BE traffic becomes larger.

In Fig. 22, when *TMR* decreases, lifetime becomes larger. It shows RT and BE-filters have different compromises between lifetime and delay. Our approach balances the energy and delay goals of the sensor network for both BE and RT traffic.

## 6. Conclusion

In this paper, we propose an energy-efficient differentiated directed diffusion (EDDD), which extends

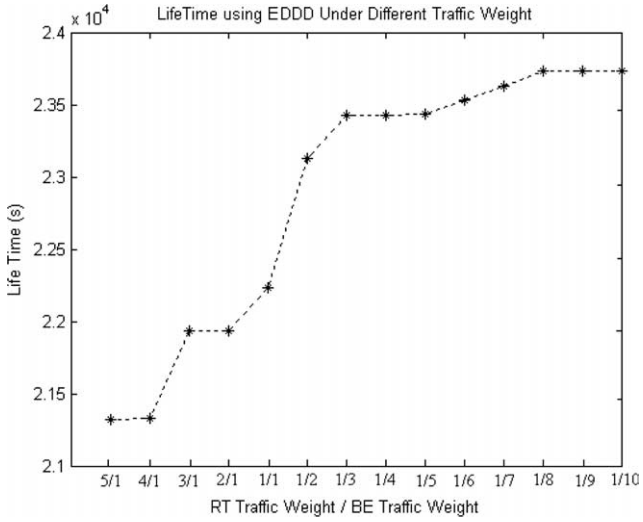


Fig. 22. Comparison LifeTimeTypeII using different TMR in EDDD.

the Directed Diffusion. EDDD provides service differentiation between real time (RT) and best effort (BE) traffic by employing new filters, namely RT filter and BE filter. Especially, the BE filter achieves global balance in energy consumption for BE traffic. The RT filter takes into consideration the hop count of a path first and then the minimum available energy along the path when it sets up the path between a sensor and a sink. Meanwhile, the BE filter puts higher priority on the minimum available energy of the path compared to the hop count. For real time traffic, a repair mechanism is employed to recover a node/link failure fast. The proposed filters and repair mechanisms of EDDD are evaluated through comprehensive simulation experiments. In the future, we will investigate how to extend EDDD for the scenarios with node/sink mobility, multiple sinks.

## Acknowledgements

This work was supported by grant No. (R01-2004-000-10372-0) from the Basic Research Program of the Korea Science and Engineering Foundation.

## References

[1] K. Akkaya, M. Younis, 'A Survey of Routing Protocols in Wireless Sensor Networks,' in the Elsevier Ad Hoc Network Journal (to appear).

[2] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks Proceedings of the Sixth Annual ACM/IEEE MobiCom'00, Boston, MA, 2000.

[3] Fabio Silva, John Heidemann, Ramesh Govindan, and Deborah Estrin. 'Directed Diffusion'. Technical Report ISI-TR-2004-586, USC/Information Sciences Institute, 2004. To appear in Frontiers in Distributed Sensor Networks, S.S. Iyengar, R.R. Brooks, eds.

[4] K. Akkaya, M. Younis, Energy-aware routing of time-constrained traffic in wireless sensor networks, International Journal of Communication Systems, Special Issue on Service Differentiation and QoS in Ad Hoc Networks 17 (6) (2004) 663–687.

[5] K. Akkaya and M. Younis, 'Energy and QoS aware Routing in Wireless Sensor Networks,' in the Cluster Computing Journal, Special Issue on Ad Hoc Networks (to appear).

[6] K. Sohrabi, et al., Protocols for self-organization of a wireless sensor network, IEEE Personal Communications 7 (5) (2000) 16–27.

[7] SPEED: A stateless protocol for real-time communication in sensor networks, Proceedings of International Conference on Distributed Computing Systems, Providence, RI (2003).

[8] B. Karp, H.T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks Proceedings of the ACM International Conference on Mobile Computing and Networking, ACM, Boston, Mass., USA, 2000. pp. 243–254.

[9] Yan Yu, Ramesh Govindan, Deborah Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report TR-01-0023, University of California, Los Angeles, Computer Science Department, 2001.

[10] F. Stann, J. Heidemann, Rmst: Reliable data transport in sensor networks Proceedings of the First International Workshop on Sensor Net Protocols and Applications, USC/Information Sciences Institute, IEEE, Anchorage, Alaska, USA, 2003. pp. 102–112.

[11] F. Zhao, J. Shin, J. Reich, Information-driven dynamic sensor collaboration, IEEE Signal Processing Magazine 61–72 (2002).

[12] D. Estrin, et al., Next Century Challenges: Scalable Coordination in Sensor Networks Proceedings of International Conference on Mobile Computing and Networks (MobiCOM'99), Seattle, WA, 1999.

[13] G.J. Pottie, W.J. Kaiser, et al., Wireless integrated network sensors, Communications of the ACM 43 (5) (2000) 51–58.

[14] K. Sohrabi, et al., Protocols for self-organization of a wireless sensor network, IEEE Personal Communications 7 (5) (2000) 16–27.

[15] W. Heinzelman, et al., Adaptive protocols for information dissemination in wireless sensor networks, Proceedings of Fifth Annual ACM/IEEE MobiCom'99, Seattle, WA (1999).

[16] Jason Hill, Mike Horton, et al., The platforms enabling wireless sensor networks, Communications of the ACM, vol. 47(6), pp. 61–46.

[17] Alec Woo, Sam Madden, Ramesh Govinda, Networking Support for Query Processing in Sensor Network, Communications of the ACM, vol. 47(6), pp. 47–52.

[18] W. Choi, P. Shah, S.K. Das, A framework for energy-saving data gathering using two-phase clustering in wireless sensor networks, Proceedings of Mobile and Ubiquitous Systems (Mobiquitous): Networking and Services, Boston (2004).

[19] R. Shah, J. Rabaey, Energy aware routing for low energy Ad Hoc sensor networks, Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Orlando, FL (2002).

[20] K. Wu, Y. Gao, F. Li, and Y. Xiao, Lightweight Deployment-Aware Scheduling for Wireless Sensor Networks, To appear in ACM/Kluwer MONET Journal, Special Issue on Energy Constraints and Lifetime Performance in Wireless Sensor Networks.

[21] M.L. Sichitiu, Cross-Layer Scheduling for Power in Wireless Sensor Networks, Proceedings INFOCOM'04, Hong Kong, China (2004).

[22] X. Hong, M. Gerla, H. Wang, L. Clare, Load balanced, energy aware communications for Mars sensor networks, IEEE Aerospace Conference Proceedings 3 (2002) 3.1109–3.1115.

[23] S.-C. Huang, R.-H. Jan, Energy-aware, load balanced routing schemes for sensor networks, Proceedings of Tenth International Conference on Parallel and Distributed Systems (ICPADS 2004) (2004) 419–425.

[25] W. Ye, J. Heidemann, D. Estrin, An energy efficient MAC protocol for wireless sensor networks, Proceedings INFOCOM'02 (2002).

[26] L.M. Feeney, M. Nilsson, Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, Proceedings INFOCOM'01 3 (2001) 1548–1557.

**Min Chen** was born in Lin Chuan, China, on Dec. 1980. He received BS, MS and PhD degree from Dept. of Electronic Engineering, South China University of Technology, in 1999, 2001 and 2004 respectively. He is currently a post-doctoral researcher in Multimedia&Mobile Communications Lab., School of Computer Science and Engineering, Seoul National University.

**Taekyoung Kwon** is an assistant professor in Multimedia & Mobile Communications Lab., School of Computer Science and Engineering, Seoul National University He received his PhD, MS, and BS degrees in computer engineering from Seoul National University in 2000, 1995, and 1993, respectively. He was a visiting student at IBM T. J. Watson Research Center in 1998 and a visiting scholar at the University of North Texas in 1999. His recent research areas include radio resource management, wireless technology convergence, mobility management, and sensor network.

**Yanghee Choi** received BS in electronics engineering from Seoul National University, MS in electrical engineering from Korea Advanced Institute of Science, and Doctor of Engineering in computer science from École Nationale upérieure des Telecommunications (ENST) in Paris, in 1975, 1977 and 1984 respectively. Before joining the School of Computer Engineering, Seoul National University in 1991, he has been with Electronics and Telecommunications Research Institute (ETRI) during 1977–1991, where he served as a director of Data Communication Section and Protocol Engineering Center. He was a research student at Centre National d'Étude des Telecommunications (CNET), Issy-les-Moulineaux, during 1981–1984. He was also a Visiting Scientist to IBM T.J. Watson Research Center for the year 1988–1989. He is now leading the Multimedia Communications Laboratory in Seoul National University. He is also a director of Computer Network Research Center in Research Institute of Advanced Computer Technology (RIACT). He was an editor-in-chief of Korea Information Science Society journals. He was a chairman of the Special Interest Group on Information Networking. He has been an associate dean of research affairs at Seoul National University. He is now the President of Open Systems and Internet Association of Korea. His research interest lies in the field of multimedia systems and high-speed networking.