An Energy-efficient Multi-candidate Greedy Routing Scheme in Wireless Sensor Networks

Dang Tu Nguyen¹, Wook Choi², Minh Thiep Ha¹, and Hyunseung Choo¹

¹School of Information and Communication Engineering, Sungkyunkwan Univ., S. Korea
Email: {dangtu,thiepha,choo}@skku.edu
²Department of Computer Science and Engineering, Hankuk Univ. of Foreign Studies, S. Korea
Email: twchoi@hufs.ac.kr

Abstract—Sleep and wake-up scheduling of sensor nodes is an efficient solution to prolong the network lifetime. However, existing scheduling algorithms may significantly decrease the number of active nodes so that the network may be intermittently-connected. In such networks, traditional geographic routing protocols are inappropriate to obtain low latency routes due to route discovery and data forwarding latency. In this paper, we propose a novel multi-candidate greedy routing (MGR) scheme that makes the best effort to find minimum latency routes in the sensor networks. In MGR, each source node sends an RREQ to a set of first wake-up forwarder candidates and selects a route with minimum estimated delivery latency based on their replies. Moreover, to cope with the unreliability of wireless links, we introduce an add-on algorithm for MGR, called multilevel of link quality requirement-based packet forwarding (MLRPF), which helps to increase the energy efficiency of MGR while still guaranteeing a low delivery latency. Simulation results demonstrate that MLRPF increases the routing performance of MGR significantly compared with PRRxDistance [24], DGF [6], and ODML [16] in terms of packet delivery latency and energy efficiency.

Index Terms—sleep latency, energy efficiency, multi-candidate greedy forwarding, geographic routing, wireless sensor networks

I. INTRODUCTION

Wireless sensor networks (WSN) are composed of a large number of sensor nodes and in general, they are deployed in inaccessible and hostile environments, e.g., dense jungles, battlefields, and inside phenomenon [1], [8], [9], [19]. Sensor nodes are often powered by batteries that may not be recharged. Meanwhile, many sensor network applications need to last for a long time. An intermittently-connected WSN. This work is a revised and more thorough study than the earlier version of our work [17]. In MGR, each source node sends an RREQ to a number of first wake-up forwarder candidates that are closer to the destination node. When an intermediate node receives an RREQ, it also forwards the RREQ to its neighbors. These steps are repeated until the RREQ reaches the destination node. Clearly, the first RREQ arriving at the destination goes through the smallest latency route. The destination node then responds with a route reply (RREP) to the source node. Because all nodes are scheduled to work and sleep, RREP may take a long time to travel back to the source node. Consequently, the end-to-end delivery latency which is the sum of route discovery and data transmission latencies, might become much larger. In addition, ODML has a high routing overhead. These issues worsen in the case that the working-duty cycle of nodes decreases or the network size increases.

In this paper, we propose a novel multi-candidate greedy routing (MGR) scheme that makes the best effort to find minimum latency routes in intermittently-connected WSN. This work is a revised and more thorough study than the earlier version of our work [17]. In MGR, each source node sends an RREQ to a number of first wake-up forwarder candidates that are closer to the destination node. When an intermediate node receives an RREQ, it also forwards the RREQ to its neighbors. These steps are repeated until the RREQ reaches the destination node or an SREP is sent to the source node. When the first RREQ arrives at the destination node, the destination node responds with an RREP to the source node. Among the replies from its forwarder candidates, the source node selects the route...
with minimum estimated delivery latency.

Recent empirical and theoretical studies on radio properties have shown that wireless links between low power sensor devices are extremely unreliable [23], [25], [26]. Therefore, the design of communication stacks should take into account such realistic radio link layer characteristics. Hence, to cope with the unreliability of wireless links, we also introduce an add-on algorithm for MGR – multilevel of link quality requirement-based packet forwarding (MLRPF) – which helps to increase the energy efficiency of MGR while still guaranteeing a low delivery latency. Our key contributions are:

- MGR together with MLRPF can be used with any scheduling algorithm and can be applied to various realistic situations based on the user’s need such as low delivery latency (e.g., fire alarm system, military surveillance) or high energy efficiency (e.g., habitat monitoring, environment monitoring).
- Besides RREQ and RREP, we introduce the concept of SREP that is used to eliminate the delivery latency of RREP. Consequently, MGR can improve the delivery latency of ODML by 99.67%.
- Since a source node sends an RREQ to only a small number of forwarder candidates and an intermediate node forwards the RREQ to only the first wake-up node, the routing overhead of MGR is much smaller than that of ODML.
- The use of MGR and MLRPF together can improve the energy efficiency of ODML and DGF by 18.49% and 31.72%, respectively. Moreover, those routes found by MGR change over time, thus helping to distribute the traffic load over nodes in the network.
- As a result, the network lifetime can be increased.

The remainder of this paper is organized as follows. In Section II, we discuss related work. Section III presents system models, assumptions, and definitions while Section IV describes the proposed scheme, MGR. Section V shows the performance evaluation results. Finally, we conclude the paper in Section VI.

II. RELATED WORK

Geographic routing protocols are efficient in wireless sensor networks due to their low cost in finding and storing routes. Traditional geographic routing protocols usually use a greedy forwarding mechanism [6] whereby each node forwards a packet to the neighbor that is closest to the destination. In always-awake WSN, the greedy forwarding is a good choice to find routes with a small number of hops. However, routes found by the greedy forwarding may have high delivery latency in intermittently-connected WSN. Moreover, the fact that routes do not change over time may cause some nodes to exhaust their energy faster than others.

In [7], the authors introduce the concept of dynamic switch-based forwarding (DSF) that addresses the above routing problems in extremely low duty-cycle WSN. Here, extremely low duty-cycle means that each sensor works for a very short working period (e.g., a single time slot) within a long round (e.g., 1000 slots). In DSF, for a given sink, each node stores a sequence of forwarding nodes. When a node has a data packet to send to the sink node, it forwards the packet to the first wake-up node in the sequence. If the transmission is successful, forwarding is done. Otherwise, the node fetches the next wake-up node from the sequence and tries to send the packet again. This retransmission process over a single hop continues until the sending node confirms that the packet has been successfully received by one of forwarding nodes or the sending node reaches the end of the sequence and drops the packet. In DSF, the communication model is from sensor nodes to a sink node and the idea of DSF is only efficient for extremely low duty-cycle sensor networks. Conversely, our proposed algorithm can find low latency route between any arbitrary pair of sensor nodes with any percentage of duty-cycle.

The work in [10] states that if the whole network topology and the schedules of all nodes are known, we can get optimal delivery paths in a delay tolerant network (DTN) by constructing a directed graph of nodes. For a large scale WSN with limited resources, however, this may be infeasible. In DTN, the authors only consider the problem whereby the network is partitioned due to the mobility of the mobile nodes. On the other hand, in the scheduled WSN, although the communication link is intermittently-connected, the physical topology of the network is always connected.

Lu Su et al. proposed an ODML routing algorithm to find minimum latency routes in intermittently-connected WSN [16]. In ODML, the source finds a route by sending an RREQ to its neighbors. When an intermediate node receives an RREQ, it records the latency of RREQ and updates the latency field of RREQ by adding the buffer delay of the next link to the original value. Then, it forwards the request to other neighbors except the one from which it receives the RREQ. Clearly, the first RREQ arriving at the destination goes through the minimum latency route. The destination node then unicasts an RREP back to the source along the minimum latency route. As the RREP travels back, each node along the path sets up a forward pointer to the node from which the RREP came. In addition, the arrival time of RREQ and the corresponding latency to the destination are recorded in the routing table. Because all nodes are scheduled to work and sleep, RREP may take a long time to travel back to the source node. Consequently, the end-to-end delivery latency which is the sum of route discovery and data transmission latencies, might become much larger. In addition, ODML has a high routing overhead.

The authors in [25] provided a comprehensive analysis of the root causes of link unreliability and asymmetry. In particular, they defined the packet reception rate as a function of distance. In order to cope with the wireless link unreliability in the network layer, several efforts have been made to define metrics that characterize the energy efficiency of communications [2], [4], [11], [18], [24], in which it is shown that communications over
the unreliable links could be optimized based on those metrics. However, those papers did not consider the sleep latency of sensor nodes with a certain duty cycle in scheduled WSN.

Based on a recently developed link loss model, Zamalloa et al. studied the performance of a wide array of forwarding strategies, via analysis, extensive simulations and a set of experiments on motes [24]. They found that the product of the packet reception rate and the distance improvement towards destination (PRR × Distance) is a highly suitable metric for geographic forwarding in realistic environments.

In [4], the authors introduced the concept of expected transmission count metric (ETX) that finds high-throughput paths on multi-hop wireless networks. ETX minimizes the expected total number of packet transmissions (including retransmissions) required to successfully deliver a packet to the ultimate destination. The ETX metric incorporates the effects of link loss ratios, asymmetry in the loss ratios between the two directions of each link, and interference among the successive links of a path. In contrast, the minimum hop-count metric chooses arbitrarily among the different paths of the same minimum length, regardless of the often large differences in throughput among those paths, and ignoring the possibility that a longer path might offer higher throughput.

In [18], D. T. Nguyen et al. proposed a novel multi-ACK-based data forwarding scheme to minimize the energy consumption for unnecessary data retransmissions of the single ACK-based retransmission mechanism. The next hop forwarder selection significantly affects the energy efficiency of communications. Hence, they also developed a new next hop forwarder selection metric, called effective energy consumption (EEC), which makes their proposed data forwarding scheme suitable for geographic routing protocols. Mathematical analysis and simulation results demonstrated that the multi-ACK-based data forwarding scheme can save much energy, reduce the total amount of traffic load, and significantly increase the energy efficiency of the network.

### III. Preliminaries

#### A. Notation and Assumptions

Let us summarize the notation used in this paper in Table I. Sensor network lifetime is divided into rounds of equal duration ( Timeout ). As shown in Fig. 1, a round consists of a working period (Time w ) and a sleep period. Each round is divided into time slots, and a packet is only transmitted at the start of a time slot. When a node is in the working period, it can sense the vicinity and receive/transmit packets. Otherwise, it turns off all functions except a timer to wake up. A node can wake up to transmit a packet at any time. Note that the proposed schemes work well even if the working period varies among the nodes.

In this work, we have several assumptions as follows:

- The network is locally synchronized so that a node knows when it can send a packet to its neighbors given their working schedules [7], [16], [29]. Local synchronization can be achieved by using a MAC-layer time stamping technique, as described in FTSP [28], which achieves an accuracy of 2.24µs with the cost of exchanging a few bytes of packets among neighboring nodes every 15 minutes.

- Each node has its own working schedule. It wakes up and works in its working time. We do not consider the case that nodes can invite their neighbors to wake up out of their working time [7], [16], [29].

- Nodes are aware of their location, one-hop neighbor location, and the position of the final destination and also they know the bidirectional link qualities of their neighbors which are decided by the packet reception rate [4], [6], [7], [24]. Similar to the work in [4], the values of the packet reception rate, i.e., PRR in and PRR out , can be measured using dedicated link probe packets. Each node broadcasts a fixed-size link-probing packet at an average period τ.

### Table I

**Descriptions of notation.**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T r</td>
<td>Duration of a round</td>
</tr>
<tr>
<td>T w</td>
<td>Working period of a round</td>
</tr>
<tr>
<td>PRR</td>
<td>Packet reception rate</td>
</tr>
<tr>
<td>PRR out</td>
<td>PRR of outgoing link for sending out a packet</td>
</tr>
<tr>
<td>PRR in</td>
<td>PRR of incoming link for receiving an ACK message</td>
</tr>
<tr>
<td>PRR h</td>
<td>High PRR</td>
</tr>
<tr>
<td>PRR l</td>
<td>Low PRR</td>
</tr>
<tr>
<td>PRR d</td>
<td>PRR for a distance d</td>
</tr>
<tr>
<td>p h</td>
<td>High probability</td>
</tr>
<tr>
<td>p l</td>
<td>Low probability</td>
</tr>
<tr>
<td>T d</td>
<td>Signal-to-noise ratio at a distance d</td>
</tr>
<tr>
<td>T s</td>
<td>High signal-to-noise ratio</td>
</tr>
<tr>
<td>T f</td>
<td>Low signal-to-noise ratio</td>
</tr>
<tr>
<td>K out</td>
<td>Number of packet retransmissions up to the first success</td>
</tr>
<tr>
<td>K in</td>
<td>Number of ACK message retransmissions up to the first success</td>
</tr>
<tr>
<td>E</td>
<td>Expected number of packet retransmissions up to the first success</td>
</tr>
<tr>
<td>E in</td>
<td>Expected number of ACK message retransmissions up to the first success</td>
</tr>
<tr>
<td>E</td>
<td>Expected number of packet retransmissions up to the first ACK success</td>
</tr>
<tr>
<td>R max</td>
<td>Maximum number of ACK retransmissions</td>
</tr>
<tr>
<td>R max</td>
<td>Maximum number of RREQ retransmissions</td>
</tr>
<tr>
<td>d cur − dst</td>
<td>Distance between the current node and the destination node</td>
</tr>
<tr>
<td>d nbor − dst</td>
<td>Distance between the neighboring node and the destination node</td>
</tr>
</tbody>
</table>

© 2011 ACADEMY PUBLISHER

![Fig. 1. Time round structure where T w denotes the working period and T r denotes the round.](image-url)
To avoid collision, $\tau$ is jittered by up to $\pm 0.1\tau$ per a probing packet. Because the probes are broadcasted, nodes do not acknowledge or retransmit them. Every node stores the the quality from probing packets it receives during the last $\omega$ seconds and it calculates the PRR from the sender at any time $t$ as:

$$P_{RR}(t) = \frac{\text{Count}(t - \omega, t)}{\omega/\tau}$$

where $\text{Count}(t - \omega, t)$ is the number of probes received during the window $\omega$, and $\omega/\tau$ is the number of probes that should have been received. In the case of the link $X \rightarrow Y$, this technique allows $X$ to measure $P_{RR_{in}}$ and $Y$ to measure $P_{RR_{out}}$. Because $Y$ knows it should receive a probe from $X$ every $t$ seconds, $Y$ can correctly calculate the current loss ratio even if no probes from $X$ arrive. Each probe from a node $X$ contains the number of probe packets received by $X$ from each of its neighbors during the last $\omega$ seconds. This allows each neighbor to calculate the $P_{RR_{out}}$ to $X$ whenever it receives a probe from $X$.

- The network is static and the link qualities between sensors do not change [4], [6], [7], [24]. However, in the case that node mobility makes the link qualities dynamic, nodes can use the probe packets to update the link qualities periodically.

### B. Link Layer Model

In the simulation of our work, we use a realistic link layer model introduced in [25], which is based on the log normal path loss model [14]. According to the log normal path loss model the received power ($P_r$) at a distance $d$ is a random variable in $dB$ given by:

$$P_r(d) = P_t - PL(d_0) - 10nlog_{10}\left(\frac{d}{d_0}\right) + N(0, \sigma) \quad (1)$$

where $P_t$ is the output power, $n$ is the path loss exponent (rate at which signal decays with respect to distance), $N(0, \sigma)$ is a Gaussian random variable with mean 0 and variance $\sigma^2$, and $PL(d_0)$ is the power decay for the reference distance $d_0$.

For a transmitter-receiver distance $d$, the signal-to-noise ratio ($\mathcal{T}_d$) at the receiver is also a random variable in $dB$, and it can be derived from Eq. 1:

$$\mathcal{T}_d = P_r(d) - P_n$$

$$= P_t - PL(d_0) - 10nlog_{10}\left(\frac{d}{d_0}\right) + N(0, \sigma) - P_n$$

$$= N(\mu(d), \sigma) \quad (2)$$

where $P_n$ is the noise floor and $\mu(d)$ is given by:

$$\mu(d) = P_t - PL(d_0) - 10nlog_{10}\left(\frac{d}{d_0}\right) - P_n$$

The values of the signal-to-noise ratio from Eq. 2 can be inserted on any of the available bit-error-rate expressions available in the communication literature. When Manchester encoding and non-coherent frequency shift keying (NCFSK)$^1$ modulation schemes are used, the PRR for a distance $d$ between a transmitter and a receiver becomes a random variable that is given by:

$$P_{RR_d} = (1 - \frac{1}{2} \exp^{-10\frac{\mathcal{T}_d}{\sigma^2}}) \times 8f$$

where $\mathcal{Y}$ is the signal to noise ratio in $dB$; $\rho$ is the encoding ratio ($\rho = 2$ for Manchester encoding); $f$ is the frame length in bytes.

In [25], M. Z. Zamalloa and B. Krishnamachari showed that the link qualities between sensors are divided into 3 regions: connected region, disconnected region, and transitional region. The definitions of these regions are as follows:

- **Connected region**: In the connected region links have a high probability ($> p_h$) of having high packet reception rates ($> P_{RR_h}$).
- **Disconnected region**: In the disconnected region links have a high probability ($> p_l$) of having low packet reception rates ($< P_{RR_l}$).
- **Transitional region**: The transitional region is the region between the end of the connected region and the beginning of the disconnected region; and $p_h$ and $p_l$ can be chosen as any numbers close to 1 and 0, respectively. The expressions for the beginning ($d_b$) and end ($d_e$) of the transitional region are given by:

$$d_b = 10^{\frac{P_h + \mathcal{T}_h - P_r - PL(d_0) + 2\rho}{10\sigma}}$$

$$d_e = 10^{\frac{P_l + \mathcal{T}_l - P_r - PL(d_0) + 2\rho}{10\sigma}}$$

where $\mathcal{T}_h$ and $\mathcal{T}_l$ are the SNR values in $dB$ corresponding to $P_{RR_h}$ and $P_{RR_l}$, respectively. In the simulation of our work, we set the transmission range of sensors to be $2[d_e]$ [24], [25].

### C. Energy Consumption Model

We consider only the energy consumption for packet transmissions. More specifically, we do not consider other kinds of energy consumption, e.g., energy spent in state (active/idle/sleep) transitions. For a node in the transmitting/receiving state, the power consumption, $P$, can be obtained by:

$$P = I \times V$$

where $I$ denotes the current consumption; $V$ is the supply voltage. Given $P$, the energy consumption, $e$, in one state can be obtained by:

$$e = P \times T$$

where $T$ is the time duration spent on that state. Simulations in this study use the power model used in the Mica2 hardware platform to measure the energy consumption. Table II shows the Mica2 power model [15].

---

$^1$NCFSK radios were chosen because we use the radio model of CC1000 equipped mica2 motes in the simulation [24], [25].
D. Probabilistic Model for Link Quality Requirement Analysis

Previous works frequently used a data packet retransmission mechanism to increase the packet delivery ratio [4], [24]. In the retransmission mechanism [27], a node sends a packet to a next hop and waits for an ACK message. If the next hop receives the packet, it responds with an ACK message to the sender node. After a certain timeout duration, if the sender node does not receive the ACK message, it retransmits the same data packet. If the next hop receives the packet, it responds with an ACK message to the sender node. After a certain timeout duration, if the sender node does not receive the ACK message, it retransmits the same data packet. These steps are repeated until the sender node receives an ACK message or the number of retransmissions exceeds the threshold. In this work, we also use the ACK-based retransmission mechanism to increase the reliability.

We assume the value of \( K_{\text{out}} \) as a random variable which follows Geometric random distribution [13], i.e., depending on the link quality between two sensor nodes, \( K_{\text{out}} \) varies. Therefore, given a \( PRR_{\text{out}} \), the probability for \( K_{\text{out}} = k \) is:

\[
P[K_{\text{out}} = k] = (1 - PRR_{\text{out}})^{k-1} PRR_{\text{out}}
\]

For the random variable \( K_{\text{out}} \), we use the expected value of \( K_{\text{out}} \), denoted by \( E[K_{\text{out}}] \), which is given by:

\[
E[K_{\text{out}}] = \sum_{k=0}^{\infty} k P[K_{\text{out}} = k] = \sum_{k=0}^{\infty} k (1 - PRR_{\text{out}})^{k-1} PRR_{\text{out}} = \frac{1}{PRR_{\text{out}}}
\]

The standard deviation of \( K_{\text{out}} \) is given by:

\[
\sigma_{\text{out}} = \sqrt{\frac{1 - PRR_{\text{out}}}{PRR_{\text{out}}}}
\]

Similarly, the expected value of \( K_{\text{in}} \), denoted by \( E[K_{\text{in}}] \), is given by:

\[
E[K_{\text{in}}] = \frac{1}{PRR_{\text{in}}}
\]

and the standard deviation of \( K_{\text{in}} \) is given by:

\[
\sigma_{\text{in}} = \sqrt{\frac{1 - PRR_{\text{in}}}{PRR_{\text{in}}}}
\]

E. Basic Definitions

Definition 1: Sleep Latency. When a node has a data packet to send, if all of its neighbors are in the sleep period, it has to wait until they wake up. The waiting time from the moment when a packet is available for sending to the time when the packet is successfully sent out is called the sleep latency. The sleep latency is much longer than other kinds of latency, e.g., processing delay, transmission delay, and propagation delay. Thus, we only consider the sleep latency in this paper.

Definition 2: Delivery Latency. Let \( R_{S,D} \) be a route between node \( S \) and node \( D \). At time \( t \), \( S \) has a packet to send to the destination node \( D \). The delivery latency of the route \( R_{S,D} \) at time \( t \) is the summation of the sleep latency of all the nodes on this route, which is spent on delivering the packet to the destination node \( D \).

Definition 3: Estimated Delivery Latency. Let \( R_{S,D} \) be a route between node \( S \) and node \( D \). A packet is delivered from \( S \) to an intermediate node \( I \) with the accumulated delivery latency \( AL \). Then the estimated delivery latency \( EL \) is given by:

\[
EL = AL + AL \times d_{I,D}
\]

where \( d_{I,D} \) is the distance between the intermediate node \( I \) and the destination node \( D \); \( d_{S,I} \) is the distance between the source node \( S \) and the intermediate node \( I \).

Definition 4: Ending Time. Assume the working period of an arbitrary node \( E \) is from time \( t_1 \) to \( t_2 \). Then \( t_2 \) is called the ending time of node \( E \).

Definition 5: Expected Stuck Time. Let \( R_{S,D} \) be a route between node \( S \) and node \( D \). At time \( t \), \( S \) can send packets to \( D \) via the route \( R_{S,D} \). At time \( t + 1 \), however, \( S \) cannot send packets to \( D \) using the same route due to that an intermediate node on the route enters into a sleep period. Then \( t \) is called an expected stuck time of the route \( R_{S,D} \). As shown in Fig. 2, at time 10, \( S \) can send packets to \( B \) via the route \( S-A-B \). However, at time 11, node \( A \) enters into a sleep period; thus, \( S \) cannot send data to \( B \) at time 11. Therefore, time 10 is called an expected stuck time.

Lemma: Suppose \( t \) is an expected stuck time of the route \( R_{S,D} \). Then, \( t \) occurs iff the state of at least one

**Table II**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Current</th>
<th>Mode</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx (-20 dBm)</td>
<td>7.1 mA</td>
<td>Tx (-5 dBm)</td>
<td>7.1 mA</td>
</tr>
<tr>
<td>Tx (-19 dBm)</td>
<td>3.7 mA</td>
<td>Tx (0 dBm)</td>
<td>8.5 mA</td>
</tr>
<tr>
<td>Tx (-15 dBm)</td>
<td>5.2 mA</td>
<td>Tx (+4 dBm)</td>
<td>11.6 mA</td>
</tr>
<tr>
<td>Tx (-8 dBm)</td>
<td>5.4 mA</td>
<td>Tx (+8 dBm)</td>
<td>17.4 mA</td>
</tr>
<tr>
<td></td>
<td>6.5 mA</td>
<td>Tx (+10 dBm)</td>
<td>21.5 mA</td>
</tr>
</tbody>
</table>

**Fig. 2.** An illustration of packet forwarding procedure using MGR.

© 2011 ACADEMY PUBLISHER
node in the route $R_{S,D}$ changes from working period to sleep period at time $t+1$.

**Proof:** Assume no nodes in the route $R_{S,D}$ enter into sleep period at time $t+1$. At time $t$, $S$ can send packets to the destination $D$, i.e., at that time all the nodes in the route $R_{S,D}$ are in active state. At time $t+1$, no nodes enter into sleep period, i.e., they are still in active state. Therefore, $S$ can still send packets to $D$ at time $t+1$. Consequently, $t$ cannot be called an expected stuck time. This is contrary to the hypothesis of the claim and hence the claim.

An expected stuck time of the route $R_{S,D}$ occurs when the state of at least one node in the route $R_{S,D}$ switches from working period to sleep period. Therefore, the earliest ending time of nodes’ working periods on the route $R_{S,D}$ must be tracked to detect an expected stuck time.

### IV. PROPOSED SCHEME: MULTI-CANDIDATE GREEDY ROUTING

#### A. Overview

In this section, we describe the proposed MGR scheme that makes the best effort to find the minimum latency routes in intermittently-connected WSN. In MGR, when a source node $S$ has a packet to send, it selects a set of first wake-up nodes from its one-hop neighbors that are closer to the destination node (e.g., nodes 2, 5, and 8 in Fig. 3). $S$ then sends an RREQ to these candidate nodes. When an intermediate node receives the RREQ and if an expected stuck time occurs, it sends an SREP that contains the expected stuck time and the estimated delivery latency, back to the source node. Otherwise, it forwards the RREQ to only a first wake-up neighbor that is closer to the destination node. These steps are repeated until the RREQ arrives at the destination node or an expected stuck time occurs. When the first RREQ arrives at the destination node, the destination node responds with an RREP to the source node. Based on the replies (SREP and RREP) from neighbors (e.g., nodes 2, 5, and 8 in Fig. 3), $S$ selects the route with minimum estimated delivery latency (e.g., the solid line starting from node 8 in Fig. 3). $S$ then sends the data packet to the last node of the selected route (e.g., node $I$). After receiving the data packet from $S$, node $I$ becomes a new source node and the above steps are repeated until the data packet arrives at the destination node. As shown in Fig. 3, $I$ receives one SREP and one RREP; therefore, $I$ sends the data packet to the destination node via the route RREP traveled (the solid line from node $I$ to node $D$).

As shown in Fig. 3, let us explain why the source node $S$ needs to send the RREQ to other nodes besides the first wake-up node. The sleep latency of the first wake-up node is smaller than those of the second and third wake-up node. However, the delivery latency of the path containing the second or third wake-up node may be smaller than that of the path containing the first wake-up node since every intermediate node on all routing paths always forwards the RREQ to its first wake-up neighbor. Therefore, in order to have a higher probability for obtaining the route with the smallest delivery latency, the source node $S$ needs to send the RREQ to the second and third wake-up nodes.

#### B. Algorithm Description

MGR routing process includes two different algorithms, one for the source node and the other for intermediate nodes. Fig. 4 and 5 show the pseudo-codes of the routing process at a source node and at an intermediate node, respectively. We describe the routing process in three sub-procedures: 1) RREQ-sending procedure of source node, 2) routing procedure of intermediate node, and 3) reply-processing procedure of source node.
Intermediate Node Routing Procedure

1: Forwarder ← select forwarder
2: RREQ’s delivery latency ← RREQ’s delivery latency + forwarder’s sleep latency
3: if current node’s ending time < RREQ’s earliest ending time
4: RREQ’s earliest ending time ← current node’s ending time
5: end if
6: if RREQ’s earliest ending time < arrival time + forwarder’s sleep latency
7: Calculate the estimated delivery latency
8: Send SREP back to source node
9: else
10: Send RREQ to the forwarder
11: end if

Fig. 5. Intermediate node routing procedure pseudo code.

1) RREQ-sending Procedure of Source Node: Suppose that a source node $S$ has a packet to send to a destination node $D$ at time $t$. As shown in Fig. 4, the source node selects a set of first wake-up nodes from its one-hop neighbors, which are closer to the destination node $D$, as its forwarder candidates (Line 1). Initially, $S$ sets the earliest ending time of the minimum latency route as its ending time (Line 2). In each iteration for all the candidates (Lines 4-13), before sending an RREQ to a forwarder, $S$ sets the earliest ending time of the RREQ as that of the minimum latency route (Lines 5-6) and the delivery latency of the RREQ as the sleep latency of the forwarder (Line 7). $S$ then sends an RREQ to the forwarder (Line 12) when this forwarder wakes up.

2) Routing Procedure of Intermediate Node: As shown in Fig. 5, when an intermediate node $I$ receives an RREQ, it selects a first wake-up node from its one-hop neighbors that are closer to the destination node $D$, as its forwarder (Line 1). Then $I$ adds the sleep latency of its forwarder to the delivery latency of the RREQ (Lines 2-3). If $I$’s ending time is earlier than the earliest ending time in the RREQ, the earliest ending time in RREQ is replaced with $I$’s ending time (Lines 4-6). After the replacement, if the earliest ending time is earlier than the time at which $I$ can send the RREQ to its forwarder (Line 7), the earliest ending time becomes an expected stuck time of the current route. Then $I$ calculates the estimated delivery latency (Line 8) and responds with an SREP to the source node (Line 9). Otherwise, $I$ sends the RREQ to its next-hop forwarder (Line 11). These steps are repeated until the RREQ arrives at the destination node or an expected stuck time occurs. When the first RREQ arrives at the destination node $D$, $D$ replies with an RREP to the source node $S$ using the route RREQ discovered. Other RREQs that arrive at $D$ later on are dropped.

3) Reply-processing Procedure of Source Node: As shown in Fig. 4, if $S$ receives an RREP from a forwarder candidate (i.e., this RREP traversed through the minimum latency route), $S$ forwards the data packet to the destination node via this forwarder candidate and stops sending RREQ (Lines 15-17). The data packet is then sent through the path via which the RREP traversed. In the case that $S$ receives an SREP (Line 18), $S$ checks whether this SREP is the first one or not. If this SREP is the first one, $S$ chooses the route of this SREP as its minimum latency route (Lines 19-20). More specifically, $S$ replaces its minimum latency route with the new route if the estimated delivery latency of the new route is smaller than that of the minimum latency route (Lines 21-23). As shown in Fig. 3, assumes that $S$ receives the first reply from node 2, i.e., the current minimum latency route of $S$ is null, thus $S$ sets the minimum latency route as the route containing node 2. After that, $S$ receives a reply from node 8. Assumes that the estimated delivery latency of the route containing node 8 is smaller than that of the route containing node 2, so $S$ replaces the minimum latency route with the route containing node 8.

$S$ continues sending RREQ and waiting for replies until the earliest ending time of the minimum latency route expires, or it finishes sending RREQ to all of its forwarder candidates, or it receives an RREP (Lines 8-10: stop sending RREQ, Lines 17 and 27: stop waiting for replies). Finally, $S$ forwards the data packet to the last node of the minimum latency route (Line 28) which becomes a new source node. The above steps are repeated until the data packet reaches the destination node. Note that in the case that a route is discovered by RREP, the last node is the destination node.

C. Algorithm Walk-through

We use Fig. 2 to illustrate the walk-through of our proposed MGR algorithm. $S$ has a data packet available at time 5. Among its one-hop neighbors, $S$ selects $A$ and $E$ as its forwarder candidates. The RREQ from $S$ is then forwarded along two paths: $SA$ and $SE$. We explain this example in four steps: 1) routing procedure on $SA$ path, 2) routing procedure on $SE$ path, 3) reply-processing procedure, and 4) advantage of using SREP.

1) Routing Procedure on $SA$ Path: $S$ sends an RREQ to $A$ at $t_5$, with the delivery latency of the RREQ set to 0 (i.e., $A$ is active at $t_5$) and the earliest ending time set to 15 (ending time of $S$). After receiving the RREQ from $S$, $A$ forwards the RREQ to $B$ at $t_5$, with the delivery latency still set to 0 (i.e., $B$ is in its working period at $t_5$) and the earliest ending time set to 10 (ending time of $A$ and it is earlier than 15, the earliest ending time of the current route). $B$ can only forward the RREQ to $C$ at $t_{11}$, thus time 10 becomes an expected stuck time. So, $B$ calculates the estimated delivery latency by using Eq. 3 and sends an SREP back to the source node $S$. The accumulative delivery latency up to node $B$ on the route $S$-$A$-$B$-$C$ is $6$ ($t_1 + t_5$) and the estimated delivery latency is $16$ ($6 + \frac{6}{60} * 100$, assuming that $d_{S,B} = 60$ and $d_{B,D} = 100$).

2) Routing Procedure on $SE$ Path: Similar to the routing procedure on $SA$ path, $S$ sends an RREQ to $E$ at $t_5$, with both the delivery latency of the RREQ set to 0 (i.e., $E$ is active at $t_5$) and the earliest ending time set to 15 (ending time of $S$). After receiving the RREQ from $S$, $E$ has to wait until $t_{20}$ to forward the RREQ to $F$ since $F$ wakes up at $t_2$. Therefore, $E$ updates the delivery latency.
to be 15 \((t_{20} - t_5)\) and the earliest ending time to be 10
(ending time of \(E\) and it is earlier than the earliest ending
time of the current route, 15). \(E\) detects an expected stuck
time, since it can only forward the RREQ to \(F\) at \(t_{20}\) (note
that \(t_{20}\) is later than the earliest ending time \(t_{10}\) of
the route \(S-E\). \(E\) uses Eq. 3 to calculate the estimated delivery
latency and sends an SREP back to the source node \(S\). The
accumulative delivery latency up to node \(E\) on the route
\(S-E-F\) is 15 and the estimated delivery latency up to the
destination node \(D\) is 60 \((15 + \frac{30}{40} + 120,\) assuming that
d_{S,E} = 40 and d_{E,D} = 120).

3) Reply-processing Procedure: After receiving two
SREPs from \(A\) and \(E\), \(S\) selects the route \(S-A-B-C\) as
its minimum latency route since the estimated delivery
latency 16 of the route \(S-A-B-C\) is smaller than that of
the route \(S-E-F\) 60, and sends a data packet to \(C\) using
\(S-A-B-C\) route. The data packet reaches \(C\) at time \(t_11\), i.e.,
the delivery latency is 6 \((11-5)\). \(C\) then becomes a new
source node and repeats the same steps as \(S\) does. Since
the destination node \(D\) is \(C\)'s neighbor and is in active
state, \(C\) directly forwards the data packet to \(D\). Therefore,
the final delivery latency is 6.

4) Advantage of Using SREP: As shown in Fig. 6, if
SREP is not used, similar to the previous scenario, \(S\) sends
an RREQ to \(A\) and \(E\) at \(t_5\), with the delivery latency of
RREQ set to 0. The RREQ reaches \(B\) at \(t_5\), \(B\) then delivers
the RREQ to \(C\) at \(t_{11}\). After receiving the RREQ from \(B,
\(C\) forwards the RREQ to the destination node \(D\) at \(t_{11}\),
thus the delivery latency of the RREQ is 6 \((t_{11} - t_6)\).
After receiving RREQ, \(D\) sends an RREP to \(C\) at \(t_{11}\), \(C\)
then sends the RREP to \(B\) at \(t_{11}\), too. However, \(A\) is in
sleep period at this time. Thus, \(B\) has to wait until the
start of the next round to forward the RREP to \(A\), i.e., the
delivery latency becomes 25 \((t_6 + t_{30} - t_{11}\), where 30 is
the time duration of one round in the working schedule). \(A\)
also has to wait until time \(t_5\) to send the RREP to \(S\),
so that the delivery latency is 30 \((t_{25} + t_5 - t_0)\).

Finally, \(S\) sends the data packet to \(D\) using the route \(S-
A-B-C-D\). The data packet reaches \(D\) at \(t_{11}\), thus the final
delivery latency is 36 \((t_{30} + t_{11} - t_5)\). In conclusion, using
SREP together with RREP helps data packets overcome
expected stuck times of routing paths, thus eliminating the
delivery latency of RREP.

Fig. 6. An illustration of packet forwarding procedure without using
SREP.

372 JOURNAL OF NETWORKS, VOL. 6, NO. 3, MARCH 2011

MLRPF Algorithm

1: for each level of link quality requirement \(LR\)
2:     Forwarder list ← Ø
3: for each neighbor of the current node
4:     if \(d_{S,r,LR} < d_{S,LR}\) and \(PRR_{r,LR} \) satisfy \(LR\)
5:         Add this neighbor to the forwarder list
6:     end if
7: end for
8: Forwarder ← first wake-up node of the forwarder list
9: MGR
10: if send the RREQ to the forwarder successfully
11:     exit the loop
12: end if
13: end for

Fig. 7. Pseudo code of the multilevel of link quality requirement based
packet forwarding (MLRPF) algorithm.

D. Multilevel of Link Quality Requirement-based Packet
Forwarding (MLRPF) Algorithm

As mentioned above, the purpose of MGR is to find
routes with the smallest delivery latency. However,
the wireless links between sensor devices in realistic envi-
rornents are unreliable. Therefore, MGR may not be
energy-efficient in such environments. To mitigate this
issue, we introduce an add-on algorithm for MGR which
helps to increase the energy efficiency of MGR while
guaranteeing a low delivery latency. As shown in Fig. 7, a
sender node select its forwarder candidates based on two
criteria: i) a candidate must be closer to the destination
node (Line 4) and ii) the link qualities of both directions
between the sender node and a candidate must satisfy
a link quality requirement (Line 4). The sender node
selects its forwarder as the first wake-up node among
these candidates. The remaining steps are the same as in
MGR. The sender node then forwards the RREQ to the
forwarder. After a maximum number of retransmissions
\(RT_{RREQ}\), if the RREQ transmission still fails, the sender
node repeats the above steps using a higher link quality
requirement. This process is repeated until the RREQ
transmission succeeds or the highest level of link quality
requirement is reached (Lines 10–12).

The key point of MLRPF is to use a multilevel of
link quality requirements in selecting forwarders. Why
do we need to use a multilevel of link quality require-
ments? When using only one level of requirement, if
the requirement is high, a sender node can find its
forwarder with good link qualities. That is, it can save
transmission energy thus helps to increase the energy
efficiency. However, the number of forwarder candidates
is small. Therefore, the sender node has less chance to
find a forwarder with a small sleep latency among the
candidates, i.e., the delivery latency of MGR may be high.
In contrast, if the requirement is low, the delivery latency
is low, but the energy efficiency is also low. Thus, by using
a multilevel of requirements, we can increase the energy
efficiency of MGR while still guaranteeing a low delivery
latency. A sender node starts with an acceptably low link
quality requirement, i.e., it has more chance to find a
forwarder with a small sleep latency and has an acceptable probability of sending out the RREQ successfully. If the RREQ transmission succeeds, the link quality between the forwarder and the sender node tends to be good. That is, MLRPF satisfies two criteria at the same time – low delivery latency and high energy efficiency. On the other hand, if the RREQ transmission fails, the sender node repeats the process using a higher level of link quality requirement. The problems here are how to select the initial level of the link quality requirement and how to set a higher requirement if the previous ones fail.

The number of RREQ retransmissions between a sender and a forwarder until the sender receives an ACK message from the forwarder can be given by:

\[ K = K_{in} \times K_{out} \]

The expected value of \( K \) can be calculated by:

\[ E[K] = E[K_{in}] \times E[K_{out}] \]

Since \( K_{in} \) and \( K_{out} \) fluctuate around \( E[K_{in}] \) and \( E[K_{out}] \) with the standard deviations \( \sigma_{in} \) and \( \sigma_{out} \), respectively, the maximum expected value of \( K \) can be given by:

\[ E[K]_{\text{max}} = (E[K_{in}] + \sigma_{in}) \times (E[K_{out}] + \sigma_{out}) \]

Therefore, given a maximum number of RREQ retransmissions – \( RT_{RREQ} \), to ensure the sender can receive an ACK message from the forwarder, we use three levels of link quality requirement as follows:

1) Low expected requirement:

\[ E[K] \leq RT_{RREQ} \]

2) Mid expected requirement:

\[ E[K]_{\text{mid}} = \frac{(E[K] + E[K]_{\text{max}})}{2} \leq RT_{RREQ} \]

3) High expected requirement:

\[ E[K]_{\text{max}} \leq RT_{RREQ} \]

Obviously, the more RREQ retransmissions, the more energy sensor nodes have to consume. Since the nodes use a multilevel of link quality requirements in selecting forwarder candidates, they have many chance to forward the RREQ successfully. Therefore, for each level of requirements, nodes should use a small value of \( RT_{RREQ} \) so that they can reduce the number of unnecessary RREQ retransmissions.

Algorithm Walk-through: We use an example in Fig. 8 to illustrate MLRPF algorithm. In this example, the sender node \( S \) has 5 neighbors (\( C_1, \ldots, C_5 \)) which are closer to the destination node and \( RT_{RREQ} = 3 \). The link qualities between \( S \) and these neighbors are shown in Table III. Assume an RREQ arrives at \( S \) at time point 12. First, \( S \) uses the low expected requirement to select its forwarder candidates. As shown in Table III, nodes \( C_2, C_3, C_4, \) and \( C_5 \) satisfy the requirement and thus become \( S \)'s forwarder candidates. Since node \( C_4 \) has the smallest sleep latency, \( S \) selects \( C_4 \) as its forwarder and forwards the RREQ to \( C_4 \). Assume that the RREQ transmission fails, so \( S \) uses the mid expected requirement to select its forwarder candidates. Similarly, \( S \) selects \( C_5 \) as its forwarder and forwards the RREQ to \( C_5 \). Assume the RREQ transmission succeeds, then \( S \) stops sending RREQ.

V. PERFORMANCE EVALUATION

In this simulation study, we use a simulator built in Java to evaluate the performance of the proposed scheme – MGR with the add-on algorithm, MLRPF – under different network sizes and duty cycles. We compare it with DGF, ODML, PRRxDistance, and the minimum latency route. As mentioned above, MLRPF significantly increases the energy efficiency and success rate of MGR. Therefore, to evaluate the contributions of MLRPF, we run simulations with networks having a realistic link layer as in [24], [25]. We used the following metrics to evaluate the performance:

- **Delivery latency**: The amount of time taken to deliver a data packet from a source node to a destination node.
- **Energy efficiency**: Number of bits successfully delivered to the destination node for each unit of energy spent by the network in communication events.

![Fig. 8. An example MLRPF algorithm.](image-url)
- **Success rate**: Percentage of packets sent by the source node that reach the destination node.

A. Simulation Environment

The minimum latency route is constructed by flooding data packets to the destination node. The delivery latency of the minimum latency route is the value of the first data packet arriving at the destination node. In the simulation, sensor nodes are randomly deployed; the transmission range of sensor node is fixed at 30m; and the round time \( T_r \) is set to 500 time slots. For varying network sizes, the number of nodes increases from 100 to 1000 with a fixed node density\(^2\). The sensor-deployed area increases proportionally to the number of nodes from \( 35m \times 35m \) to \( 124m \times 124m \). The duration of the working period \( T_w \) is 100 time slots. For the simulation of different duty cycles, 100 sensor nodes are deployed in \( 124m \times 124m \) network area. The node duty cycle varies from 10% to 50% of the round time. That is, \( T_w \) varies from 50 time slots to 250 time slots. Table IV summarized the parameters used in the simulation. For each value of duty cycle and network size, simulation is run 1000 times and we average them to get the mean value. In each run, a pair of source and destination nodes are randomly chosen. The source node then randomly select a time slot in its working period for the packet arrival time. Note that we do not consider the energy consumed for routing overhead when calculating energy efficiencies of ODML and MGR.

B. Simulation Results

1) **Delivery latency**: Fig. 9 shows the influence of network size on the performance of schemes in terms of end-to-end delivery latency. PRRxDistance and DGF perform the worst, while MGR is the best and close to the minimum latency route. The delivery latency of ODML is much smaller than the values of PRRxDistance and DGF, but about 2 times greater than that of MGR. The reason is that it may take a long time for RREP to travel back to the source node due to expected stuck times in the minimum latency routes. Consequently, the total latency of ODML which is the sum of RREQ latency, RREP latency, and data latency becomes much greater. For MGR, however, during sending an RREQ, if an expected stuck time occurs, the current node responds the source node with a SREP. Therefore, MGR does not suffer the delay time caused by expected stuck times. As a result, the delivery latency of MGR is small. For example, when the number of nodes is 1000, the delivery latencies of MGR, PRRxDistance, DGF, and OMDL are 599, 2359, 2288, and 1196, respectively. That is, MGR increases the performance by 293.82% (\( (2359 - 599)/599 \approx 2.9382 \)) compared with PRRxDistance, 281.97% (\( (2288 - 599)/599 \approx 2.8197 \)) compared with DGF, and 99.67% (\( (1196 - 599)/599 \approx 0.9967 \)) compared with ODML.

Fig. 10 reports the end-to-end delivery latency under different node duty cycles. Similar to the previous sce-
delivery latency of ODML is very high since there are
is the best and close to the minimum latency route. It
nario, PRRxDistance and DGF are the worst and MGR
is worth noting that when node duty cycles are low, the
delivery latency of ODML is very high since there are
more expected stuck times in the minimum latency routes.

2) Energy efficiency: Fig. 11 shows the energy effi-
ciencies of four schemes under different network sizes. PRRxDistance is the best at all network sizes, while DGF is the worst because DGF does not consider link quality when selecting forwarders. MGR is better than ODML and DGF since MGR uses the add-on algorithm, MLRPF, to select forwarders. For instance, when the number of nodes is 1000, the energy efficiencies of MGR, DGF, and OMDL are 8.14, 6.18, and 6.87, respectively. That is, MGR increases the performance by 31.72% ((8.14 – 6.18)/6.18 ≈ 0.3172) compared with DGF and 18.49% ((8.14 – 6.87)/6.87 ≈ 0.1849) compared with ODML.

The energy efficiencies of schemes under different node
duty cycles are shown in Fig. 12. The energy efficiencies
increase because nodes with good link qualities have
more chance to be selected as forwarders. The energy efficiencies of PRRxDistance and DGF, in contrast, are
the same for all duty cycles.

3) Success rate: Fig. 13 reports the success rates of
schemes under different network sizes. Similar to the
case of energy efficiency, PRRxDistance is the best at
all network sizes, while DGF is the worst because DGF
does not consider link quality when selecting forwarders.
MGR is better than ODML and DGF. For example,
when the number of nodes is 1000, the success rates of
MGR, DGF, and OMDL are 0.808, 0.474, and 0.754,
respectively. That is, MGR increases the reliability by
70.46% ((0.808 – 0.474)/0.474 ≈ 0.7046) compared
with DGF and 7.16% ((0.808 – 0.754)/0.754 ≈ 0.0716)
compared with ODML.

Fig. 14 shows the success rates of schemes under
different node duty cycles. Similar to the case of energy
efficiency, the success rates of ODML and MGR increase
when node duty cycles increase; the success rates of
PRRxDistance and DGF, in contrast, are the same for all
duty cycles.

VI. CONCLUSION

In this paper, we propose a novel multi-candidate greedy routing (MGR) scheme that makes the best effort to find minimum latency routes in intermittently-connected WSN. However, MGR may not be energy-efficient in realistic applications of WSN since MGR does not consider the unreliability nature of wireless links. Hence, we introduce an add-on algorithm for MGR, called multilevel of link quality requirement-based packet forwarding (MLRPF), which helps to increase the energy efficiency of MGR while still guaranteeing a low delivery latency. The proposed scheme can be used with any scheduling algorithm and applied to various realistic situations based on the user’s need such as low delivery latency (e.g. fire alarm system, military surveillance) or high energy efficiency (e.g. habitat monitoring, environment monitoring). Simulation results demonstrate that MGR with MLRPF increases the routing performance significantly compared with PRRxDistance, DGF, and ODML, in terms of packet delivery latency and energy efficiency.

ACKNOWLEDGMENT

This research was supported in part by MKE and MEST, Korean government, under ITRC NIPA-2010-(C1090-1021-0008), WCU NRF (No. R31-2008-000-10062-0) and PRCP (2010-0020210) through NRF, respectively. Dr. Choo is the corresponding author.

REFERENCES


Author Biographies

Dang Tu Nguyen received a BS degree (with honors) from University of Science – Ho Chi Minh City, Vietnam National University, Vietnam in 2007. From 2007 to 2008, he was with the Faculty of Information Technology, University of Science of Ho Chi Minh City (VNU), Vietnam. From 2009, he joined the Networking Laboratory, Sungkyunkwan University, Korea. His research interests include routing protocols, energy efficiency, reliability, and cross-layer optimization in wireless sensor networks; system and networking applications; and wireless communication protocols.
Wook Choi is an assistant professor of Computer Science and Engineering Department at Hankuk University of Foreign Studies. Before joining this university, he worked as a senior engineer in Samsung Electronics and as a principal engineer in Microsoft, respectively. He finished his Ph.D. at the University of Texas at Arlington (UTA) in 2005. His research interests include wireless mesh networks, ad hoc and sensor networks, mobile and pervasive computing, multi-radio access protocols, ubiquitous health care, ubiquitous sensor networks.

Minh Thiep Ha received a BS degree (with honors) from Vietnam National University of Ho Chi Minh City in 2007. From 2008, he is a Master’s Degree candidate in Electrical and Computer Engineering from Sungkyunkwan University, Korea. His research interests include clustering, routing, and localization algorithms in wireless adhoc/sensor networks.

Hyunseung Choo received the B.S. degree in mathematics from Sungkyunkwan University, Korea, in 1988, the M.S. degree in computer science from the University of Texas at Dallas, in 1990, and the Ph.D. degree in computer science from the University of Texas at Arlington, in 1996. From 1997 to 1998, he was a Patent Examiner at the Korean Industrial Property Office. Since 1998, he has been with the School of Information and Communication Engineering, Sungkyunkwan University, and is an Associate Professor and Director of the Convergence Research Institute. Since 2005, he has been the Director of the Intelligent HCI Convergence Research Center (eight-year research program) supported by the Ministry of Knowledge Economy (Korea) under the Information Technology Research Center support program supervised by the Institute of Information Technology Assessment. He is Vice President of the Korean Society for Internet Information (KSII). He has published over 250 papers in international journals and refereed conferences. His research interests include wired/wireless/optical embedded networking, mobile computing, and grid computing. Dr. Choo has been Editor-in-Chief of the Journal of Korean Society for Internet Information for three years and Journal Editor of the Journal of Communications and Networks, the ACM Transactions on Internet Technology, the International Journal of Mobile Communication, Springer-Verlag Transactions on Computational Science Journal, and Editor of the KSII Transactions on Internet and Information Systems since 2006. He is a member of the ACM, IEEE, and IEICE.