Itinerary Planning for Energy-Efficient Agent Communications in Wireless Sensor Networks

Min Chen, *Senior Member, IEEE*, Laurence T. Yang, *Member, IEEE*, Taekyoung Kwon, *Associate Member, IEEE*, Liang Zhou, *Member, IEEE*, and Minho Jo, *Member, IEEE*

***Abstract*—Compared with conventional wireless sensor networks (WSNs) operating based on the client–server computing model, mobile agent (MA)-based WSNs can facilitate agent-based data aggregation and energy-efficient data collection. In MA systems, it has been known that finding the optimal itinerary of an MA is nondeterministic polynomial-time hard (NP-hard) and is still an open area of research. In this paper, we consider the impact of both data aggregation and energy efficiency in itinerary selection. We first propose the Itinerary Energy Minimum for First-source-selection (IEMF) algorithm. Then, the itinerary energy minimum algorithm (IEMA), which is the iterative version of IEMF, is described. This paper further presents a generic framework for the multiagent itinerary planning (MIP) solution, i.e., the determination of the number of MAs, allocating a subset of source nodes to each agent and itinerary planning for each MA. Our simulation results have demonstrated that IEMF provides higher energy efficiency and lower delay, compared with existing single-agent itinerary planning (SIP) algorithms, and IEMA incrementally enhances IEMF at the cost of computational complexity. The extensive experiments also show the effectiveness of MIP algorithms when compared with SIP solutions.**

***Index Terms*—Data aggregation, energy efficiency, mobile agent (MA), wireless sensor networks (WSNs).**

Manuscript received September 18, 2010; revised January 17, 2011; accepted March 1, 2011. Date of publication April 5, 2011; date of current version September 19, 2011. This work was supported in part by the National Agenda Project of the Korea Research Council of Fundamental Science and Technology; by The Ministry of Knowledge Economy, Korea, under the Information Technology Research Center support program supervised by the National IT Industry Promotion Agency under Grant NIPA-2011-(C1090-1111-0004); by the Natural Science and Engineering Research Council of Canada; and by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No.2011-0009454). The review of this paper was coordinated by Dr. T. Taleb.

M. Chen is with the School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea (e-mail: minchen@ieee.org).

L. T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada (e-mail: ltyang@stfx.ca).

T. Kwon (Corresponding author) is with the School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea (e-mail: tkkwon@snu.ac.kr).

L. Zhou is with the College of Communication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: liang.zhou@ieee.org).

M. Jo is with the College of Information and Communications, Korea University, Seoul 136-713, Korea (e-mail: minhojo@korea.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2011.2134116

I. Introduction

THE APPLICATION-SPECIFIC nature of wireless sensor networks (WSNs) requires that sensor nodes have various capabilities for diverse applications. It would be impractical to store all of the programs needed in the local memory of embedded sensors to run every possible application, due to the tight memory constraints. A mobile agent (MA) is a special kind of software that migrates among sensor nodes to autonomously carry out task(s) in response to changing conditions in the network environment to achieve the objectives of the agent dispatcher (i.e., the sink node). The use of MAs to dynamically deploy new applications in WSNs has been proven to be an effective method to address this challenge [1]–[7].

In [1], the agent design in WSNs is decomposed into four components, i.e., architecture, itinerary planning, middleware system design, and agent cooperation. Among the four components, itinerary planning determines the order of source nodes to be visited during agent migration, which has a significant impact on the energy performance of the MA systems. It has been shown that finding an optimal itinerary is nondeterministic polynomial-time hard (NP-hard) [3]. Therefore, heuristic algorithms are generally used to compute itineraries with suboptimal performance.

In [2], two simple heuristic algorithms are proposed: 1) the local closest first (LCF) scheme, which searches for the next node with the shortest distance to the current node, and 2) the global closest first (GCF) scheme, which searches for the next node closest to the sink. These two schemes only consider the spatial distances between sensor nodes and, thus, may not be energy efficient in many cases. Mobile Agent-based Directed Diffusion (MADD) [4] is similar with LCF but differs in selecting the first source node. MADD selects the farthest source node from the sink as the first source. In [3], a genetic algorithm (GA) is proposed to produce near-optimal itinerary results in WSNs. Its main components include an encoding mechanism, crossover and mutation operations, and an evaluation function.

Some previous schemes (e.g., LCF, GCF [2], and GA [3]) are all based on the assumption of high redundancy among the sensory data, which can be fused into a single data packet with a fixed size. This implies that a perfect data aggregation model is used. The assumption limits the application scope of WSNs. To address this limitation, we present a general data aggregation model, which facilitates a wide range of applications.

In this paper, we focus on designing energy-efficient itinerary planning algorithms while relaxing the aforementioned

TABLE I
NOTATION

Symbol	Definition
l_{data}	the size of raw sensory data at a source node.
S_{proc}	the size of processing code.
l_{ma}^{0}	the size of mobile agent when dispatched from the sink.
l_{ma}^{k}	the agent size when it leaves the kth source.
r	the reduction ratio by agent assisted local processing.
l_{rd}	reduced data payload after local processing.
ρ	data aggregation ratio, a measure of redundancy elimination.
n	the number of source nodes to be visited.
V	the set of source nodes to be visited, where $\|V\| = n$.
p	the starting point (node) of the agent.
t	the sink node.
S	the resulting sorted source visiting set based on a planned itinerary (PI).
$S[1]$	the first source node in S.
$S[k]$	the kth source node in S.
SIP(u, V, t)	a single agent-based itinerary planning algorithm, which returns S.
$f(u, V, t)$	the next-source-selection function, which returns the next source to be visited.
$d(u, v)$	the distance between nodes u and v,
H_{k-1}^{k}	the hop counts between source k-1 to source k.
$E_{k-1}^{k}(l_{\text{ma}})$	the communication energy cost during a mobile agent roams from source k-1 to source k with agent size l_{ma}.
E_I	the communication energy cost of an itinerary.
$I(u \rightarrow v)$	the itinerary segment from the uth source $S[u]$ to the vth source $S[v]$.
$E_{I(u \rightarrow v)}$	the energy cost of itinerary segment $I(u \rightarrow v)$.
κ	the iterative number in IEMA algorithm.
IEMA(κ)	IEMA with the iterative number of κ.
$\eta_{n=N}(\kappa)$	the integrated performance of IEMA(κ) in the scenario with N source nodes.
$\alpha(\kappa)$	the performance incremental percentage of IEMA(κ) compared to LCF algorithm.

assumption. We first propose an Itinerary Energy Minimum for First-source-selection (IEMF) algorithm, which extends LCF by considering the estimated communication cost. In IEMF, the impact of both data aggregation and energy efficiency is taken into account to obtain an energy-efficient itinerary. The scheme is quite general in the sense that relies on no specific network architecture (e.g., [2] assumes a cluster-based networking environment).

We then propose an Itinerary Energy Minimum Algorithm (IEMA), which is an iterative version of IEMF. During each iteration, IEMA selects the best node according to IEMF as the next source to visit among the remaining set of source nodes. We show that, with more iterations, the suboptimal itinerary can be progressively improved and that the major reduction in average energy consumption is achieved for the first few iterations. We can thus trade off between energy efficiency and computational complexity based on specific application requirements.

Although our proposed IEMF and IEMA approaches exhibit higher performance in terms of energy efficiency, compared with the existing solutions, the limitation of utilizing a single agent to perform the whole task makes the algorithm unscalable with a large number of source nodes to be visited.

For large-scale sensor networks, with many nodes to be visited, single-agent-based data collection may exhibit two pitfalls.

1) Large delay: Extensive delay is needed when a single agent works for networks comprising hundreds or more sensor nodes.
2) Unbalanced load: There are two kinds of unbalancing problems while using a single agent. First, in the perspective of the whole network, the traffic load is put on the nodes along the single flow. Therefore, sensor nodes traversed along the itinerary will quickly deplete energy than other nodes. Second, from the perspective of the itinerary, the agent size (since it includes data) continuously increases while it visits source nodes, and so, the agent transmissions will consume more energy in its itinerary back to the sink node.

In this paper, we further study multiagent itinerary planning (MIP) algorithms to address the preceding issue and present a generic framework for the design of a MIP algorithm. The notation used in this paper is given in Table I. The rest of this paper is organized as follows: The single-agent itinerary planning (SIP) problem is formulated in Section II. We present IEMF and IEMA in Section III. The MIP problem is explained

in Section IV. We analyze the computational and space complexities in Section V. Our simulation studies are reported in Section VI. Section VII concludes this paper and presents the future work.

II. Single-Agent Itinerary Planning Problem

A. Data Aggregation Model

The degree of correlation between sensory data from two sensor nodes is closely related to the distance between them, as well as a particular application scenario. Typically, closely located sensors are very likely to generate data with high redundancy. In densely populated WSNs, data aggregation becomes a very important function for energy conservation, which reduces the redundancy in sensor data and, thus, decreases the volume of data to be transmitted. Many traditional data aggregation schemes exploit a specific network structure (e.g., a cluster-based network [2] or forming a data aggregation tree structure [9]). In MA-based WSNs, an MA visits the source nodes one after the other. At each source node, the MA collects sensory data and then performs data aggregation and removes any existing redundancy, depending on the data aggregation function. There is no need for a specific topology structure. However, the data aggregation and energy performance are highly dependent on the order in which the source nodes are visited, i.e., the itinerary.

Consider an MA dispatched by the sink node to collect data from n source nodes. Let l_{proc} be the size of the MA processing code, S_{head} be the size of the agent packet header, and l_{ma}^0 be the agent size when it is first dispatched by the sink node. Then, we have $l_{\mathrm{ma}}^0 = l_{\mathrm{proc}} + S_{\mathrm{head}}$. Let $r \in [0,1)$ be the reduction ratio in sensory data by agent-assisted local processing and l_{data} be the size of raw data at a source node. The reduced data payload collected by the agent at each source, which is denoted by l_{rd}, is $l_{\mathrm{rd}} = (1-r)\cdot l_{\mathrm{data}}$. Let l_{ma}^k be the agent size when it leaves the kth source $(1 \le k \le n)$. Since there is no data aggregation at the first source, we have $l_{\mathrm{ma}}^1 = l_{\mathrm{ma}}^0 + l_{\mathrm{rd}}$.

When the agent visits the second source node, it begins to perform data aggregation to reduce the redundancy between the data collected at the current source and the data it has carried. Let $\rho \in [0,1]$ denote the data aggregation ratio, which is a measure of the compression performance. The MA size, after it leaves the second source node, is $l_{\mathrm{ma}}^2 = l_{\mathrm{ma}}^0 + l_{\mathrm{rd}} + (1-\rho)l_{\mathrm{rd}}$, and so forth. For the sake of simplicity, we assume that r, ρ, and l_{data} are identical for every source.[1] After visiting the kth source node, we have

$$\begin{aligned} l_{\mathrm{ma}}^k &= l_{\mathrm{ma}}^{k-1} + (1-\rho)l_{\mathrm{rd}} \\ &= l_{\mathrm{ma}}^0 + [1 + (k-1)(1-\rho)]\, l_{\mathrm{rd}}. \end{aligned} \tag{1}$$

After visiting all the n source nodes, the MA's size is l_{ma}^n in the range of $[l_{\mathrm{ma}}^0 + l_{\mathrm{rd}}, l_{\mathrm{ma}}^0 + nl_{\mathrm{rd}}]$, depending on ρ. The lower bound $l_{\mathrm{ma}}^0 + l_{\mathrm{rd}}$ corresponds to a perfect data aggregation model $(\rho = 1)$, where multiple sensory data are compressed into a single and fixed one [2], [3], whereas the upper bound $l_{\mathrm{ma}}^0 + nl_{\mathrm{rd}}$ corresponds to the case of no data aggregation performed at the MA $(\rho = 0)$.

B. Generic SIP Algorithm

Algorithm 1 SIP(u, V, t): a SIP algorithm with a greedy approach

begin
notation
u is the starting point of an agent.
V denotes the set of source nodes to be visited by the agent.
t is the sink node.
S is the sequence of n nodes in an itinerary, which is an array.
$f(p, T, t)$ is a function that returns the next source node determined by a particular next-source-selection algorithm.
initialization
$T \leftarrow V$.
$p \leftarrow u$.
for $i = 1$ to n **do**
$S[i] \leftarrow f(p, T, t)$.
$p \leftarrow S[i]$.
$T \leftarrow T - \{S[i]\}$.
end for
Return S.

We state our assumptions and define SIP algorithm in this section. Specifically, it is assumed that one node (i.e., the sink node) is more powerful than the others for computing itinerary planning algorithms. The issues of node mobility and node failure are not considered in this paper.[2] We assume that the set of source nodes to be visited is predetermined. In addition, the location information of the source nodes is also available at the sink node [2], [3]. Under these assumptions, we actually consider static itinerary planning [1], where an itinerary is chosen based on the location information of the source nodes. Note that these are the general assumptions made in the references discussed in this paper.

Let n represent the number of source nodes to be visited and V denote the set of source nodes to be visited by an MA. In addition, let u and t be the starting point and ending point of an agent, respectively. The ending point t is always the sink node. Although, usually, u is set to the sink node, it can also be set to one of the source nodes when defining an iterative algorithm (see Section III-C). We define a generic SIP algorithm as a

[1] The case of heterogeneous r_i, ρ_i, and $S_{\mathrm{data},i}$ at each source node i can be easily handled in a similar fashion.

[2] An MA may deal with unexpected failures of source nodes. The MA then changes the visiting order of sensors by skipping the crashed nodes.

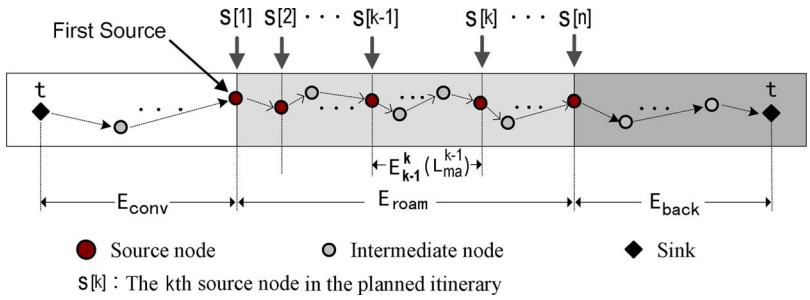


Fig. 1. Estimating communication cost of a tentatively planned itinerary in IEMF.

function SIP(u, V, t). Specifically, we design a SIP algorithm with two components.

1) Next-source-selection function, which is denoted by $f(p, T, t)$. Given a source list T, the current starting point p, and the sink node t, it returns the best source node to be visited next, among the $|T|$ source candidates. In LCF [2], $f(p, T, t)$ finds w $(w \in T)$, such that the distance between p and w is the minimum, i.e., $d(p, w) = \min\{d(p, j)|j \in T\}$. As for GCF, $f(p, T, t)$ finds w, such that the distance between w and t is the minimum, i.e., $d(w, t) = \min\{d(j, t)|j \in T\}$.
2) SIP algorithm, which is denoted by SIP(u, V, t). Given V, a starting point u, the sink node t, and a next-source-selection function $f(p, T, t)$, it computes a planned itinerary $S = S[1]|S[2]|\cdots|S[n]$, where $|$ is concatenation of sequences. The pseudocode of a SIP algorithm is given in Algorithm 1.

III. Proposed Single-Agent Itinerary Planning Algorithms

A. Estimated Communication Cost of a Candidate Itinerary

We first show how to estimate the communication cost of a given itinerary $t|S[1]|S[2]|\cdots|S[n]|t$, which means that an agent starts from sink t and returns back to t after visiting n source nodes, as shown in Fig. 1.

Generally, the communication energy consumption for a packet transmission at a given node consists of the receiving energy, the control energy, and the transmitting energy. Let e_{ctrl} be the energy spent on control messages exchanged for a successful data transmission (e.g., acknowledgement). Let m_{rx} and m_{tx} be the energy consumption for receiving and transmitting a data bit, respectively. Let c_{tx} denote the fixed energy cost for each transmission, which is independent of the packet length. Without loss of generality, we assume that m_{tx}, m_{rx}, c_{tx}, and e_{ctrl} are the same for every node. Let l_{rx} and l_{tx} be the sizes of a received and a transmitted packet, respectively. When a node receives a packet with size of l_{rx}, after local processing, the size of a transmitted packet by this node (l_{tx}) is different. The communication energy consumption at a node (i.e., an intermediate node or a source node) can be expressed as

$$e(l_{\text{rx}}, l_{\text{tx}}) = m_{\text{rx}} \cdot l_{\text{rx}} + (m_{\text{tx}} \cdot l_{\text{tx}} + c_{\text{tx}}) + e_{\text{ctrl}}. \quad (2)$$

Multiple hops may exist between two adjacent source nodes, e.g., $S[k-1]$ and $S[k]$. Let $d(S[k-1], S[k])$ denote the distance between the two source nodes. In a dense WSN, we can estimate the hop count between $S[k-1]$ and $S[k]$ as $H^k_{k-1} = \lceil d(S[k-1], S[k])/R \rceil$, where R represents the maximum transmission range. When the agent traverses intermediate sensor nodes (not the source nodes), the agent size remains the same; its size will be increased after visiting each of the source nodes. Let E^k_{k-1} be the communication energy consumed when the MA moves from $S[k-1]$ to $S[k]$ with size l^{k-1}_{ma}. We estimate the communication energy cost as

$$E^k_{k-1} = m_p \cdot l_{\text{data}} + e\left(0, l^{k-1}_{\text{ma}}\right) + H^k_{k-1} \cdot e\left(l^{k-1}_{\text{ma}}, l^{k-1}_{\text{ma}}\right) + e\left(l^{k-1}_{\text{ma}}, 0\right) \quad (3)$$

where $m_p \cdot l_{\text{data}}$ is the data-processing energy at $S[k-1]$; $e(0, l^{k-1}_{\text{ma}})$ is the energy for $S[k-1]$ to transmit the agent; $H^k_{k-1} \cdot e(l^{k-1}_{\text{ma}}, l^{k-1}_{\text{ma}})$ is the energy consumption of intermediate sensor nodes to relay the agent between $S[k-1]$ and $S[k]$; and $e(l^{k-1}_{\text{ma}}, 0)$ is the energy consumption for $S[k]$ to receive the agent.

We divide the whole itinerary into three phases, as shown in Fig. 1.

1) Code-conveying phase, which is the phase when the processing code is conveyed to the target region, during which the MA migrates from the sink to the first source node $S[1]$. The communication energy consumption in this phase is denoted by E_{conv}, i.e., $E_{\text{conv}} = H(t, S[1]) \cdot e(l^0_{\text{ma}}, l^0_{\text{ma}})$, where $H(t, S[1])$ is the estimated hop count between the sink node and S[1].
2) Roaming phase, which starts from the time when the MA arrives at the first source node $S[1]$ to the time when it arrives at the last source node $S[n]$. The communication energy consumption in this phase is denoted by E_{roam}, where $E_{\text{roam}} = \sum_{k=2}^{n} E^k_{k-1}$.
3) Returning phase, which starts from the time when the MA starts processing at the last source node to the time when it returns to the sink. The communication energy consumption in this phase is denoted by E_{back}, i.e., $E_{\text{back}} = m_p \cdot l_{\text{data}} + e(0, l^n_{\text{ma}}) + H(S[n], t) \cdot e(l^n_{\text{ma}}, l^n_{\text{ma}})$, where $m_p \cdot l_{\text{data}}$ is the data-processing energy at the last source node $S[n]$; $e(0, l^n_{\text{ma}})$ is the energy for $S[n]$ to transmit the agent; and $H(S[n], t) \cdot e(l^n_{\text{ma}}, l^n_{\text{ma}})$ is the energy consumption of intermediate sensor nodes between the last source node and the sink node.

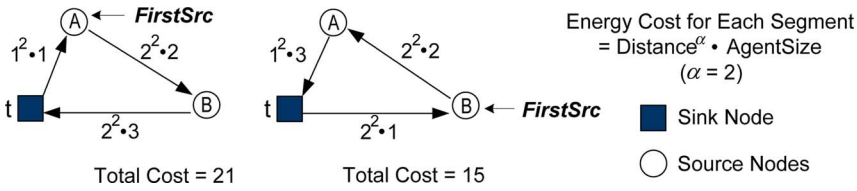


Fig. 2. Choosing different first source nodes results in different total costs. The agent size is assumed to be incremented by 1 at each source. The path-loss exponent α is set to 2.

Finally, the communication energy of a candidate itinerary is estimated by

$$E_I = E_{\text{conv}} + E_{\text{roam}} + E_{\text{back}} = E_{\text{conv}} + \sum_{k=2}^{n} E_{k-1}^{k} + E_{\text{back}}. \quad (4)$$

B. IEMF Algorithm

Fig. 2 illustrates the problem of LCF. There are three nodes in the chain of the WSN, i.e., the sink and two source nodes. Based on LCF, node A will be first visited. Assume that the distance is one between sink t and node A. The distance between t and B, and the distance between A and B are 2, respectively. Let the original agent size be 1, and assume that the agent size is increased by 1 after visiting each of the sources. We can calculate the phitinerary segment cost through multiplying the phitinerary segment distance to the power of the path-loss exponent by the current agent size. The resulting total cost of LCF when node A is visited first is 21. If the agent visits node B first, the total cost is decreased to 15. Thus, LCF performance can be improved by carefully choosing the first source node in the itinerary, which is one of the motivations for us to design the IEMF algorithm.

On the other hand, among the n source nodes in V, different algorithms may select different source nodes as $S[1]$. For example, LCF and GCF select the one that is the closest to the sink as $S[1]$; in contrast, MADD selects the farthest source node from the sink as $S[1]$. While these studies can be categorized as pure distance-based selection, IEMF selects a source node as $S[1]$ by estimating the minimum energy cost.

Specifically, the IEMF algorithm first selects an arbitrary source node v as a tentative $S[1]$. The remaining source set is denoted by $V - \{v\}$. Next, v is set as the start point, and the LCF criterion is used to determine the itinerary for the $n-1$ source nodes in $V - \{v\}$. The result of the function $LCF(v, V - \{v\}, t)$[3] is the visiting sequence for the remaining $n-1$ source nodes. Then, the entire itinerary sequence starting from the sink is obtained by $t|v|LCF(v, V - \{v\}, t)|t$.

Choosing every source in V as tentative $S[1]$ and finding the itinerary starting with that source in a round-robin fashion, we can get n different candidate itineraries and their corresponding energy costs. Among the n candidates, IEMF selects the itinerary that has the minimum energy cost. The pseudocode of IEMF is shown in Algorithm 2.

[3]For the sake of simplicity, $LCF(u, V, t)$ denotes the SIP algorithm by LCF approach, as detailed in Section IV-B.

Algorithm 2 IEMF(t, V, t): IEMF algorithm

begin
notation
IEMF(t, V, t) is the next-source-selection of IEMF.
S is a sequence of n nodes.
$|$ is the symbol to concatenate two sequences.
$LCF(v, T, t)$ is an itinerary of source nodes planned by LCF.
initialization
$tmp \leftarrow \infty$.
for each v $(v \in V)$ **do**
if $E(t|v|LCF(v, V - \{v\}, t)|t) < tmp$ **then**
$tmp \leftarrow E(t|v|LCF(v, V - \{v\}, t)|t)$.
$S[1] \leftarrow v$.
end if
end for
IEMF$(t, V, t) \leftarrow S[1]$.
$S \leftarrow S[1]|LCF(S[1], V - \{S[1]\}, t)$.
Return S.

C. IEMA Algorithm

IEMF selects the first source (or $S[1]$) as the one whose corresponding itinerary is estimated to have the smallest energy cost among n candidate itineraries. Once $S[1]$ is determined, the corresponding itinerary is actually determined by the LCF criterion [2]. In this section, we propose an iterative version of IEMF called IEMA. Compared with IEMF, in addition to $S[1]$, IEMA seeks to optimize the remaining itinerary to a certain degree.

Let κ denote the number of iterations in IEMA. Since each iteration optimizes one next hop of the itinerary, we have $\kappa \in [0, n]$. We denote IEMA with κ iterations by IEMA(κ). Specifically, LCF and IEMF are the two special cases of IEMA: LCF is the 0 iteration of IEMA, i.e., IEMA(0); and IEMF is the 1 iteration of IEMA, i.e., IEMA(1).

The pseudocode of IEMA is shown in Algorithm 3. Given κ, IEMA(κ) only optimizes the first κ source nodes using the basic IEMF method, as illustrated in Algorithm 2. Here, IEMF(u, T, t) returns the first node $S[1]$ from the result of IEMF(u, V, t).

The remaining $n - \kappa$ source nodes will be simply sorted through the LCF method. Clearly, κ provides a convenient tradeoff between energy saving and computational complexity.

Algorithm 3 IEMA(κ, t, V, t): IEMA(κ) algorithm

begin
initialization
  $u \leftarrow t$.
  $T \leftarrow V$.
for $i = 1$ to κ **do**
    $S[i] \leftarrow$ IEMF(u, T, t).
    $u \leftarrow S[i]$.
    $T \leftarrow T - \{u\}$.
end for
$S \leftarrow S[1]|S[2]|\cdots|S[\kappa]|LCF(u, T, t)$.
Return S.

D. Performance Analysis

In the following, we prove that IEMA yields the more energy-efficient itinerary after each iteration. Let $I(i \rightarrow j), 0 \leq i < j \leq n$, be the itinerary segment from the ith sorted source $(S[i])$ to the jth sorted source $(S[j])$. For example, $I(1 \rightarrow \kappa - 1) = S[1]|S[2]|\cdots|S[k-1]$, and $I(\kappa \rightarrow n) = S[k]|\cdots|S[n]$. Then, the sorted sequence $I(1 \rightarrow n)$ is also equal to $I(1 \rightarrow \kappa - 1)|I(\kappa \rightarrow n)$.

Proposition 1: Letting $E^{\text{iema}}_{I(i \rightarrow j)}(\kappa)$ denote the energy cost of $I(i \rightarrow j)$ using IEMA(κ), we have that $E^{\text{iema}}_{I(1 \rightarrow n)}(\kappa) \leq E^{\text{iema}}_{I(1 \rightarrow n)}(\kappa - 1), k \in [1, 2, \ldots, n]$.

Proof: In the case of $k = 1$, from Algorithm 2, the cost of the itinerary selected by IEMF is the minimum among all the n candidate itineraries. That is, $E^{\text{iema}}_{I(1 \rightarrow n)}(1) \leq E^{\text{iema}}_{I(1 \rightarrow n)}(0)$.

We next recursively prove the general case when $\kappa \geq 2$. From Algorithm 3, the first $\kappa - 1$ source nodes are exactly the same in both IEMA(κ) and IEMA$(\kappa - 1)$. The difference between them is in the remaining source sequences. Thus, we have

$$E^{\text{iema}}_{I(1 \rightarrow m-1)}(\kappa) = E^{\text{iema}}_{I(1 \rightarrow m-1)}(\kappa - 1) \qquad \forall m \leq \kappa. \quad (5)$$

In the κth iteration of IEMA(κ), only $n - \kappa + 1$ sources are undetermined. Since the κth iteration is the last iteration, the resulting sequence of the remaining sources is equal to IEMF$(S[\kappa - 1], V - \{S[1], S[2], \ldots, S[\kappa - 1]\}, t)$. In contrast, for IEMA$(\kappa - 1)$, since there is no IEMF operation after the $(\kappa - 1)$th iteration, $S[\kappa], \ldots, S[n]$ is equal to $LCF(S[\kappa - 1], V - \{S[1], S[2], \ldots, S[\kappa - 1]\}, t)$. Thus, $E^{\text{iema}}_{I(\kappa \rightarrow n)}(\kappa)$ is smaller than $E^{\text{iema}}_{I(\kappa \rightarrow n)}(\kappa - 1)$, according to Algorithm 2.

Based on the preceding analysis and (5), we show that the energy cost of IEMA(κ) is smaller than that of IEMA$(\kappa - 1)$, as

$$\begin{aligned} E^{\text{iema}}_{I(1 \rightarrow n)}(\kappa) &= E^{\text{iema}}_{I(1 \rightarrow k-1)}(\kappa) + E^{\text{iema}}_{I(\kappa \rightarrow n)}(\kappa) \\ &\leq E^{\text{iema}}_{I(1 \rightarrow \kappa-1)}(\kappa) + E^{\text{iema}}_{I(\kappa \rightarrow n)}(\kappa - 1) \\ &= E^{\text{iema}}_{I(1 \rightarrow \kappa-1)}(\kappa - 1) + E^{\text{iema}}_{I(\kappa \rightarrow n)}(\kappa - 1) \\ &= E^{\text{iema}}_{I(1 \rightarrow n)}(\kappa - 1). \end{aligned}$$

■

IV. Multiagent Itinerary Planning Problem

While previous studies focus on designing SIP algorithms, the use of multiple MAs for data collection is more challenging. This section will discuss the problem of MIP for a large-scale sensor network.

A. Motivation

The motivation for the MIP problem is hinted from (1) and (4), which show that the itinerary cost is a quadratically increasing function of n, thus causing the performance of the SIP algorithm to deteriorate in large-scale sensor networks. The end-to-end agent delay exhibits a similar trend as the itinerary cost. Thus, we are motivated to design a MIP algorithm that can dispatch multiple MAs, depending on the specific network parameters, such as the network size, number of source nodes, reduction ratio, data aggregation ratio, sensor data size, etc.

B. Generic MIP Algorithm

We state our assumptions and define a generic MIP algorithm in this section.

1) Most of the SIP and MIP algorithms are running at the sink, which has relatively plenty of resources in terms of energy and computation.
2) The sink node knows the geographic information of all the source nodes to be visited.

In fact, the preceding assumptions are common in most of the solutions presented for the SIP problem. The previous SIP algorithms assume that the set of source nodes to be visited by an MA is predetermined. In contrast, our MIP algorithm needs to exclusively group source nodes for multiple MAs. Finding an optimal agent number is also an NP-hard problem. Thus, we propose an iterative framework to cover all the sources by incrementing the number of MAs.

The generic MIP problem can be divided into three subproblems: 1) determination of the number of MAs; 2) allocating a subset of source nodes to each agent; and 3) itinerary planning for each MA. Among these three problems, the third one can be addressed by SIP algorithms. Thus, the main focus of MIP algorithm is to solve the first two problems.

For example, in CL-MIP [10], the visiting area of an MA is determined by the circle centered at a visiting central location (VCL). Then, the source nodes within the circular area will be assigned to the MA. The algorithm will iteratively perform until all the source nodes are allocated to MAs, and the number of MAs is equivalent to the number of iterations. By comparison, DSG-MIP [11] determines the visiting area of an MA by partitioning the whole area into directional sectors with a controllable angle. In BST-MIP [11], a minimum spanning tree is built with the sink as root and source nodes as tree nodes, and a balancing factor α is introduced to calculate of the weight of each edge of the tree. Then, the source nodes in an MA in a branch stemmed from the sink will be assigned for an MA to visit. The number of such main branches is also equal to the number of MAs.

Since finding optimal itineraries for MAs in a MIP algorithm is an NP-hard problem, all of the aforementioned heuristic algorithms introduce some adjustable parameters (e.g., radius of the circular area R in CL-MIP, expanding angle θ in DSG-MIP, and balancing factor α in BST-MIP) to facilitate a tradeoff between energy and delay while computing suboptimal itineraries.

For all the three MIP algorithms, an iterative framework can be summarized. In each iteration, a list of source nodes will be assigned to a new MA. Then, the itinerary for the MA can be planned by any SIP algorithm. If there are still-remaining sources, the aforementioned process will be repeated until all of the source nodes have been assigned to MAs. The pseudocode of the iterative MIP framework is shown in Algorithm 4.

Algorithm 4 MIP(V)

notation
V is the set of the original source nodes to be visited.
V' denotes the set of the remaining source nodes, currently.
T denotes the set of grouped source nodes at each iteration.
k denote the index of MAs.
initialization
$V' \leftarrow V$.
$k \leftarrow 0$.
loop
if V' is not empty **then**
$k \leftarrow k + 1$.
Compute T.
$V' \leftarrow V' - T$.
(phItinerary of MA k) $\leftarrow$ SIP(t, T, t).
end if
end loop
Return (Itineraries of MAs $1, 2, \ldots, k$).

V. Complexity Analysis

A. Computational Complexity

1) GCF: This essentially utilizes sorting the distances (between the sink and other sources) to compute the MA path. Its computational complexity is $O(n \log n)$ if using a comparison-based sorting algorithm (e.g., quick sort).
2) LCF: This has the computational complexity of $O(n^2)$ if the closest neighbor node is obtained by comparing the distances with the remaining nodes at each step.
3) IEMF: Intuitively, IEMF is deemed to have n times the complexity of LCF, i.e., $O(n^3)$. However, the complexity can be reduced. As shown in Table II, assume that we have a network with five source nodes (i.e., node A, B, C, D, and E). We first sort the source nodes by their distances to a test source node, which corresponds to a candidate of $S[1]$ in IEMF(t, V, t) and is given in the first column in Table II. The computational complexity of calculating each row in Table II is $O((n-1)\log(n-1))$; then, the computational complexity of calculating the whole table is $O(n(n-1)\log(n-1)) = O(n^2 \log n)$. The complexity of deciding an itinerary starting with a given node by looking up the table is $O(n \log n)$. For example, an itinerary starting with node A is given as follows: A, C, D, E, B. Then, IEMF can be finished by computing the cost of n itineraries with complexity of $O(n^2 \log n)$. Thus, the final complexity of IEMF is $O(n^2 \log n) + O(n^2 \log n) = O(n^2 \log n)$.
4) IEMA: The complexity of IEMA with κ iterations is $O(\kappa \cdot n^2 \log n)$.
5) MIP: Given CL-MIP [10] as example, the computational complexities of the VCL-selection algorithm and the source-grouping algorithm are $O(n^2)$ and $O(n)$, respectively. Thus, for each iteration of our MIP algorithm, the computational complexity will depend on the SIP algorithm. For example, if an SIP (e.g., LCF) has a computational complexity of $O(n^2)$, then one iteration of the LCF-based MIP algorithm will have the computational complexity of $O(n^2) + O(n) + O(n^2) = O(n^2)$. Then, the computational complexity of our MIP algorithm is $O(m \cdot n^2)$, where m is the maximum number of MAs. By the same approach, if using IEMF as the SIP algorithm, the corresponding MIP algorithm will have the computational complexity of $O(m \cdot n^2 \log n)$.

TABLE II
Distance Orders of Other Source Nodes to Test Sources

Test Node	1	2	3	4
A	C	B	E	D
B	D	C	A	E
C	D	B	E	A
D	E	A	B	C
E	C	D	B	A

B. Space Complexity

Space complexity will determine whether a heavy-weight sink node is always needed or whether the algorithms can also operate in a homogeneous WSN that is purely peer to peer. Given the number of source nodes of n, the minimum memory with space complexity of $O(n)$ is required to store the resulting sequence of the n source nodes after performing the itinerary planning algorithm.

1) GCF: Before the algorithm executes, it only needs to store n distances (between the sink and the other n sources). Thus, the space complexity is $O(n)$.
2) LCF: The matrix of the distance between each source–sink pair or source–source pair is required prior to the execution of the algorithm, which requires memory with space complexity of $O(n^2)$. For each iteration of the algorithm, memory with $O(1)$ is needed to store the information of the next source. Thus, the space complexity is $O(n^2)$.
3) IEMF: Similarly, it requires memory with a space complexity of $O(n^2)$ to store the distance matrix as an LCF. Then, the space complexity of $O(n)$ is required to store one itinerary candidate with any of the n source nodes as the first source node. Thus, the space complexity of

TABLE III
COMPARISONS OF THE ENERGY–DELAY PRODUCTS OF LCF, IEMF, AND IEMA

κ	Scheme	$EDP_{10}(\kappa)$	$\alpha_{10}(\kappa)$	$EDP_{20}(\kappa)$	$\alpha_{20}(\kappa)$	$EDP_{30}(\kappa)$	$\alpha_{30}(\kappa)$	$EDP_{40}(\kappa)$	$\alpha_{40}(\kappa)$
0	LCF	0.2259	0%	0.3481	0%	0.4665	0%	0.6123	0%
1	IEMF	0.1957	92.9%	0.2984	91.1%	0.4013	86.1%	0.5224	83.7%
2	IEMA(2)	0.1951	93.5%	0.2971	93.5%	0.4007	86.9%	0.5212	84.8%
5	IEMA(5)	0.1945	95.5%	0.2965	94.7%	0.3991	89.0%	0.5168	88.9%
10	IEMA(10)	0.1930	100%	0.2961	95.4%	0.3989	89.3%	0.5136	91.9%
20	IEMA(20)	-	-	0.2936	100%	0.3949	94.6%	0.5117	93.7%
30	IEMA(30)	-	-	-	-	0.3908	100%	0.5093	95.9%
40	IEMA(40)	-	-	-	-	-	-	0.5049	100%

$O(n^2)$ is needed to store n candidate itineraries. Finally, the space complexity of the algorithm is $O(n^2)$.

4) IEMA: In each iteration, IEMA requires memory with space complexity of $O(n^2)$ to store the n candidate itineraries as in IEMF and memory with $O(1)$ to store the resulting next source node. Then, the memory with $O(n^2)$ can be reused for next iteration. Thus, it has the same space complexity of $O(n^2)$ as LCF and IEMF.
5) MIP: For each iteration of a typical MIP algorithm (e.g., CL-MIP [10]), the space complexity will depend on the SIP algorithm. For example, if an SIP (e.g., LCF, IEMF, or IEMA) has space complexity of $O(n^2)$, then one iteration of the MIP algorithm will have space complexity of $O(n^2)$. Then, the space complexity of the MIP algorithm is $O(m \cdot n^2)$, where m is the maximum number of MAs.

VI. PERFORMANCE EVALUATION

We implement several SIP and MIP algorithms using OPNET Modeler and perform extensive simulations. We choose a network where nodes are uniformly deployed within a field measuring 1000 m $\times$ 500 m. To verify the scaling property of our algorithms, we select a large-scale network with 800 nodes. The multiple source nodes are randomly distributed in the network. The sensor application module consists of a constant-bit-rate source, which generates a sensory data report, and the default value of its size is 1024 bits long. By default, the size of sensed (raw) data (l_{data}) is set to 2048 bits long, the size of the processing code (l_{proc}) is 1024 bits, the raw data reduction ratio (r) is 0.8, the data aggregation ratio (ρ) is 0.9, the MA accessing delay (τ) is 10 ms, and the data-processing rate (V_p) is 50 Mb/s. For consistency, we use the same energy consumption model as in [10] and [12].

We consider three performance metrics.

1) Task duration: In an SIP algorithm, it is equivalent to average end-to-end report delay, which is the average delay from the time when an MA is dispatched by the sink to the time when the MA returns to the sink. In the MIP algorithm, since multiple MAs work in parallel, there must be one agent that returns to the sink last. Then, the task duration of a MIP algorithm is the delay of that MA.
2) Average communication energy: This consists of the total communication energy consumption, including transmissions, receptions, retransmissions, overhearing, and collisions, to obtain each sensory data from all the target sources.
3) Energy–delay product (EDP): For time-sensitive applications over energy-constrained WSNs, EDP (calculated by $EDP = energy \cdot delay$) gives us a unified view. The smaller the value of EDP, the better the unified performance.

In the succeeding sections, we first compare several SIP algorithms and examine the impact of the number of source nodes on the performance of the these algorithms. Then, typically, MIP schemes are compared with a representative SIP algorithm.

A. Performance Comparison of Representative SIP Algorithms

We set the number of source nodes n to 10, 20, 30, and 40 and obtain a set of results for each case. For a given n value, we randomly choose the set of source nodes in each simulation with a different random seed.

Typically, the number of source nodes n has a big impact on the delay and energy performance; the product of delay and energy consumption is larger for bigger n value. This is because a large n means more source nodes to be visited. The MA size will be larger, and more transmissions will be made. n is a good indicator of the MA-related traffic load. From the descriptions of the algorithms, IEMA(0) is actually equivalent to LCF, and IEMA(1) is equivalent to IEMF.

Let $EDP_n(\kappa)$ denote the EDP of IEMA(κ) in the scenario with κ iterations and n source nodes $(n = 10, 20, 30, 40, \kappa \leq n)$. In addition, we define $\alpha_n(\kappa)$ as the incremental performance gain of IEMA(κ), compared with IEMA(0) (i.e., LCF), as defined in

$$\alpha(\kappa) = \frac{EDP(0) - EDP(\kappa)}{EDP(0) - EDP(n)}, \qquad 0 \leq \kappa \leq n. \tag{6}$$

Then, in Table III, $EDP_n(\kappa)$ and $\alpha_n(\kappa)$ with various n and κ are tabulated.

Since the maximum value of κ is equal to n, we mark "$-$" when κ exceeds n. Based on Table III, we have three observations.

1) $\alpha_n(\kappa)$ is monotonically increasing with κ, which confirms that IEMA(κ) always outperforms IEMA($\kappa - 1$).
2) In the first iteration, more than 80% incremental gain of EDP is obtained. However, as the iteration increases, the performance gain between two adjacent iterations becomes marginal.

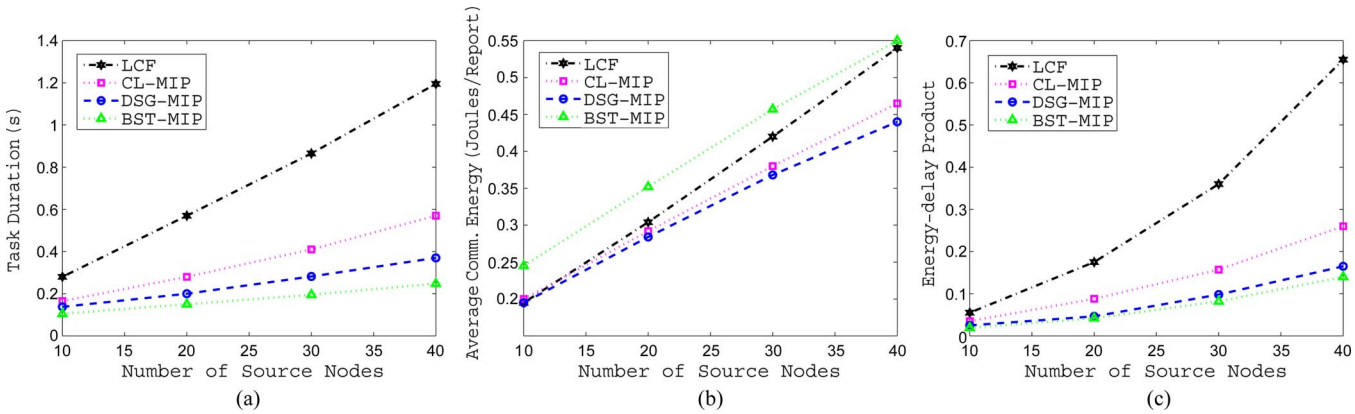


Fig. 3. Impact of the number of source nodes on performances. (a) Task duration. (b) Average communication energy (c) EDP.

3) Since most of the incremental gain is obtained in the first iteration, the simple algorithm IEMF can achieve good performance with small overhead. Generally, we can determine a suitable κ for the given application quality-of-service requirements; there is no need to have n iterations in many cases. In contrast, we find that the GA algorithm [3] achieves visible improvement only after about 100 iterations. Such fast convergence property of the proposed scheme is highly desirable.

B. Performance Comparison of MIP Algorithms and SIP With Varying Number of Source Nodes

Fig. 3(a) shows the task duration comparison among a typical SIP algorithm (i.e., LCF) and the three MIP schemes. LCF takes the longest task duration since the single MA has to visit all source nodes distributed in the network. The value of end-to-end delay for LCF grows from 0.3 to 1.2 s, along with the increasing number of source nodes. In contrast, the task durations of MIP algorithms are much lower. Fig. 3(b) illustrates the impact of the number of source nodes on energy cost. In LCF, the energy cost for LCF grows from 0.2 to 0.54 J per task while the number of source nodes increases. For the MIP solutions, BST-MIP [12] consumes more energy than LCF, whereas CL-MIP [10] and DSG-MIP [11] have higher energy efficiency. Fig. 3(c) evaluates the overall performance of the LCF and MIP schemes in terms of EDP. Due to the good performance of MIP algorithms, CL-MIP, DSG-MIP, and BST-MIP have comparable EDP values, which are much lower than that of LCF. It demonstrates the effectiveness of the MIP algorithms.

VII. Conclusion and Future Work

In this paper, we have addressed the problem of itinerary planning for MAs in dense WSNs. Based on a general data aggregation model, as well as relaxed assumptions, we have presented IEMF (an extension of LCF), which achieves improved energy performance by choosing the first source node to be visited according to the estimated communication cost. We have also presented IEMA(κ), which is an iterative version of IEMF, where the selection of the first κ source nodes is optimized based on the estimated energy costs. We have shown that the proposed schemes achieve considerable improvements in both energy saving and delay over the existing schemes, whereas IEMA(κ) provides a tradeoff between energy cost and computational complexity. On the other hand, we have considered the more challenging case of itinerary planning for multiple MAs and presented a design framework for typical MIP algorithms. Compared with SIP algorithms, MIP algorithms may achieve even lower delay and can be flexibly adaptive to network dynamics in various network scales.

References

[1] M. Chen, S. Gonzalez, and V. C. M. Leung, "Applications and design issues of MAs in wireless sensor networks," *IEEE Wireless Commun.*, vol. 14, no. 6, pp. 20–26, Dec. 2007.

[2] H. Qi and F. Wang, "Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks," in *Proc. IEEE ICC*, Helsinki, Finland, Jun. 2001, pp. 147–153.

[3] Q. Wu, N. S. V. Rao, J. Barhen, S. S. Iyenger, V. K. Vaishnavi, H. Qi, and K. Chakrabarty, "On computing mobile agent routes for data fusion in distributed sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 6, pp. 740–753, Jun. 2004.

[4] M. Chen, T. Kwon, Y. Yuan, Y. Choi, and V. Leung, "Mobile agent-based directed diffusion in wireless sensor networks," *EURASIP J. Adv. Signal Process.*, vol. 2007, no. 1, pp. 219–242, Jan. 2007.

[5] H. Qi, Y. Xu, and X. Wang, "Mobile-agent-based collaborative signal and information processing in sensor networks," *Proc. IEEE*, vol. 91, no. 8, pp. 1172–1183, Aug. 2003.

[6] M. Chen, V. Leung, S. Mao, T. Kwon, and M. Li, "Energy-efficient itinerary planning for mobile agents in wireless sensor networks," in *Proc. IEEE ICC*, Dresden, Germany, Jun. 14–18, 2009, pp. 5026–5030.

[7] M. Chen, T. Kwon, Y. Yuan, and V. Leung, "Mobile agent based wireless sensor networks," *J. Comput.*, vol. 1, no. 1, pp. 14–21, Apr. 2006.

[8] Y. Xu and H. Qi, "Distributed computing paradigms for collaborative signal and information processing in sensor networks," *Int. J. Parallel Distri. Comput.*, vol. 64, no. 8, pp. 945–959, Aug. 2004.

[9] A. Harris, R. Snader, and I. Gupta, "Building trees based on aggregation efficiency in sensor networks," *Ad Hoc Netw. J.*, vol. 5, no. 8, pp. 1317–1328, Nov. 2007.

[10] M. Chen, S. Gonzlez, Y. Zhang, and V. Leung, "Multiagent itinerary planning for sensor networks," in *Proc. QShine*, Las Palmas de Gran Canaria, Spain, 2009, pp. 584–597.

[11] M. Chen, S. Gonzalez, and V. Leung, "Directional source grouping for multi-agent itinerary planning in wireless sensor networks," in *Proc. ICTC*, Jeju Island, Korea, 2010, pp. 207–212.

[12] M. Chen, W. Cai, S. Gonzalez, and V. Leung, "Balanced itinerary planning for multiple mobile agents in wireless sensor networks," in *Proc. 2nd Int. Conf. ADHOCNETS*, Victoria, BC, Canada, 2010, pp. 416–428.

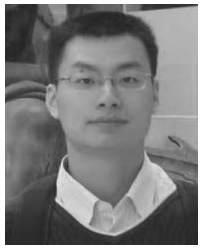
Min Chen (M'08–SM'09) has been a Postdoctoral Fellow with Seoul National University, Seoul, Korea, where he has been an Assistant Professor with the School of Computer Science and Engineering since September 2009. He has been a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancover, BC, Canada, for three years.

Dr. Chen is a symposia cochair for the 2012 IEEE International Conference on Communications.

Laurence T. Yang (M'97) is currently with the Department of Mathematics and Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. His research includes high-performance computing and networking, embedded systems, and ubiquitous/pervasive computing and intelligence. His research is supported by the National Sciences and Engineering Research Council, Canada, and the Canada Foundation for Innovation.

Taekyoung Kwon (A'00) received the B.S., M.S., and Ph.D. degrees in computer engineering from Seoul National University, Seoul, Korea, in 1993, 1995, and 2000, respectively.

He was a Visiting Student with the IBM T. J. Watson Research Center, Yorktown Heights, NY, in 1998 and a Visiting Scholar with the University of North Texas, Denton, in 1999. He is currently an Associate Professor with the Multimedia and Mobile Communications Laboratory, School of Computer Science and Engineering, Seoul National University. His recent research interests include radio resource management, wireless-technology convergence, mobility management, and wireless sensor networks.



Liang Zhou (M'09) received the Ph.D. degree in electronic engineering from both Ecole Normale Supérieure, Cachan, France, and Shanghai Jiao Tong University, Shanghai, China, in March 2009.

From 2009 to 2010, he was a Postdoctoral Researcher with Ecole Nationale Superieure de Techniques Avancees, ParisTech, Paris, France. Since October 2010, he has been an Alexander von Humboldt Research Fellow with Munich University of Technology, Munich, Germany.

Minho Jo (M'07) is a Brain Korea Professor with the College of Information and Communications, Korea University, Seoul. His current interests are in the areas of cognitive radio, wireless sensor networks, radio-frequency identification, wireless mesh networks, security in communication networks, wireless body-area networks, energy-efficient (green) communications, and ubiquitous and mobile computing.

Dr. Jo is the Executive Director of the Korean Society for Internet Information (KSII). He is the Founding Editor-in-Chief and the Chair of the Steering Committee of the *KSII Transactions on Internet and Information Systems*. He serves as an Editor of *IEEE Network*.