Energy Efficient Cooperative Computing in Mobile Wireless Sensor Networks

Zhengguo Sheng, *Member, IEEE*, Chinmaya Mahapatra, *Student Member, IEEE*, Victor C. M. Leung, *Fellow, IEEE*, Min Chen, *Senior Member, IEEE*, and Pratap Kumar Sahu, *Member, IEEE*

Abstract—Advances in future computing to support emerging sensor applications are becoming more important as the need to better utilize computation and communication resources and make them energy efficient. As a result, it is predicted that intelligent devices and networks, including mobile wireless sensor networks (MWSN), will become the new interfaces to support future applications. In this paper, we propose a novel approach to minimize energy consumption of processing an application in MWSN while satisfying a certain completion time requirement. Specifically, by introducing the concept of cooperation, the logics and related computation tasks can be optimally partitioned, offloaded and executed with the help of peer sensor nodes, thus the proposed solution can be treated as a joint optimization of computing and networking resources. Moreover, for a network with multiple mobile wireless sensor nodes, we propose energy efficient cooperation node selection strategies to offer a tradeoff between fairness and energy consumption. Our performance analysis is supplemented by simulation results to show the significant energy saving of the proposed solution.

Index Terms—Edge and cloud computing, mobile wireless sensor networks, cooperation

1 Introduction

Cloud computing [1], [2], [3] has been proposed as an efficient and cost effective way of providing highly scalable and reliable infrastructures and services. The key idea of cloud computing is to create a pool of shared, visualized, dynamically configurable and manageable resources across computing devices, networks, servers and data centers, which can deliver on demand services to users over the Internet [4]. However, existing cloud computing models are designed for traditional web applications, rather than future Internet applications running on various mobile and sensor nodes. Particularly as we go to the era of Internet of Things (IoT) with one trillion endpoints worldwide, that creates not only a real scalability problem but the challenge of dealing with complex clusters of endpoints, rather than dealing with individual endpoints. Moreover, public clouds, as they exist in practice today, are far from the idealized utility computing model, since it makes their network distance too far from many users to support highly latency-sensitive applications. This is particularly true for applications that are developed for a particular provider's platform and running in data centers that exist at singular points in space.

In contrast to the cloud, edge computing [5], which runs generic application logic on resources throughout networks, including routers and dedicated computing nodes, has attracted a lot of attention and been considered as a complementary of cloud computing to distribute intelligence in networks, and allows its resources to perform low-latency processing near the edge while latency-tolerant, large-scope aggregation can still be efficiently performed on powerful resources in the core of the cloud.

In a simple, topological sense, edge computing works in conjunction with cloud computing, optimizing the use of their resource. Users can subscribe services via the cloud computing platform which can offer diverse storage and computation capabilities from both central server and distributed nodes, respectively. Today, with the development of wireless technologies and embedded processor, the edge computing capability can be largely extended to a broad range of wireless devices, such as smartphone and wireless sensor nodes, to support flexible services.

Particularly, wireless sensor nodes, which are commonly with a radio transceiver and a microcontroller powered by a battery, as well as diverse novel sensors, are in the creation of imaginative pervasive computing applications. We have already witnessed that smartphones, such as iPhone and Android, can replace a normal desktop or a server running a dual core processor for computation [6], [7]. Moreover, the recent advancement of small size and low cost sensor platforms such as WRTnode[1] and Arduino [8], which can offer

- *Z. Sheng is with the School of Engineering and Informatics, University of Sussex, Brighton BN1 9RH, United Kingdom. E-mail: z.sheng@sussex.ac.uk.*
- *C. Mahapatra and V.C.M. Leung are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada. E-mail: {chinmaya, vleung}@ece.ubc.ca.*
- *M. Chen is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: minchen2012@hust.edu.cn.*
- *P.K. Sahu is with the Department of Computer Science and Operation Research, University of Montreal, Montreal, QC H3T 1J4, Canada. E-mail: sahupk@iro.umontreal.ca.*

Manuscript received 5 Jan. 2015; revised 18 May 2015; accepted 6 July 2015. Date of publication 17 July 2015; date of current version 7 Mar. 2018. Recommended for acceptance by P. Corcoran. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCC.2015.2458272

1. WRTnode is an open source development board with a wireless controller based on OpenWrt. It is suitable for speech/video recognition, Open CV and Machine Learning, etc. http://wrtnode.com/.

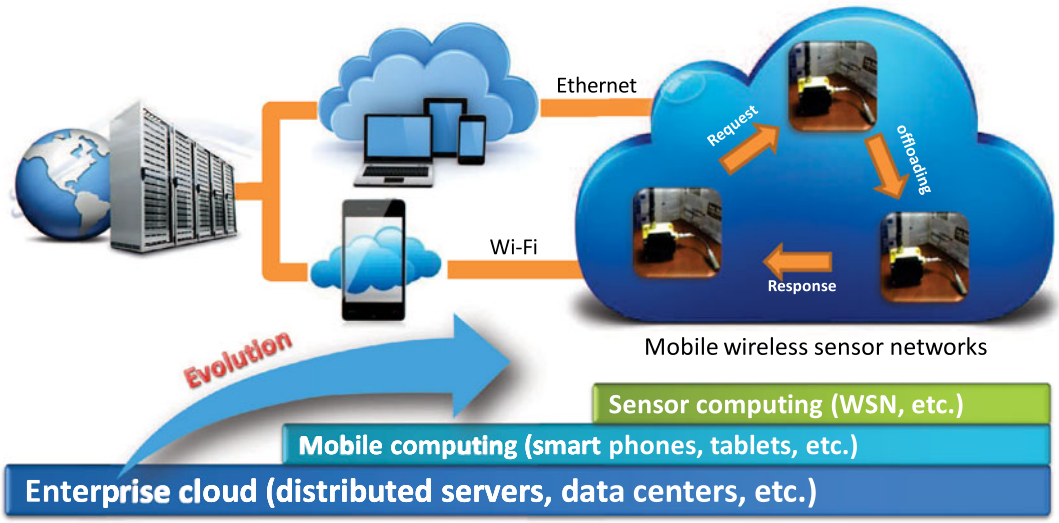


Fig. 1. Evolution from cloud computing to edge computing.

CPU clock speeds of up to 600 MHz and low power IEEE 802.11/15.4 radios, are capable of connecting external sensors (e.g., camera sensor, thermal sensor, heartbeat sensor, air pollution sensor, etc.) to support attractive lightweight sensing applications in various domains, such as environmental monitoring [9], social networking [10], healthcare [11] and transportation [12], etc. In addition to the development of efficient software [13] and communication protocols, e.g., Constrained Application Protocol (CoAP) [14], [15], the formed wireless sensor networks can truly enable the newly emerging sensor-as-a-service (SaaS) paradigm. Another motivation to leverage the sensor-based computing infrastructure is that IoT applications are rapidly developed in a number of areas, ranging from personal devices to industrial automations. The existing cloud infrastructure can benefit from significant energy savings by offloading tasks to powerful sensor nodes. We believe that such an emerging dissemination of wireless sensor networks and cloud computing can bring new opportunities of sensor and cloud integration, which will facilitate clients to not only monitor and collect data from the environment, but also execute and output sensor applications using their own processing capabilities. Fig. 1 gives our prediction of cloud computing evolution to a large scale and distributed sensor-as-a-service infrastructure.

There have been a number of technical challenges to build such a sensor-based computing infrastructure. In particular, the biggest hurdle to harness sensor nodes for computing is the battery life. In this paper, we investigate fundamental characteristics of mobile wireless sensor networks (MWSN) computing in terms of energy efficiency and propose a novel approach to optimize total energy consumption of processing an application, while satisfying a certain completion deadline requirement. Specifically, we firstly introduce the concept of cooperative computing which encourages single nodes to share their resources cooperatively such that a virtual resource pool can be constructed. Fig. 1 also shows an example of the cooperative computing serving client service requests from outside world. Moreover, by assuming the application profile with a limited size of input data and a completion deadline, the proposed solution can jointly consider computation and communication costs as a whole, and optimally partition, offload and execute workload between sensor nodes to boost energy efficiency of the edge computing. Based on these analytical results, we further propose energy efficient cooperation strategies for resource allocation in networks with multiple sensor nodes.

The following summarizes our contributions and key results:

- We introduce mathematical models to characterize application profile, computation and communication energy. Especially, the derived closed-form solution of energy consumption is highly related to the input data size and completion deadline. Moreover, by considering the mobility nature of MWSN, the proposed solution can ensure the energy performance with minimum transmission time.
- We propose an optimal partition to minimize the total energy consumption required by local and remote sensor nodes in cooperative computing under static channel model to satisfy a given deadline requirement. Furthermore, an offloading decision rule is defined to indicate the best computing strategy. Moreover, under the optimal partition, our analysis shows that the required energy consumption of a remote node (helper) is always smaller than that of a local node, a result which lays a foundation to encourage the cooperative behaviors which means that the helping part only needs to spend relatively small amount of energy than the one seeking help from others.
- By utilizing the optimal results, we propose energy efficient cooperation node selection strategies to achieve fairness and maximal energy saving in a multi-node environment, and analyze node's "willingness" to cooperate when selfish and unselfish natures are imposed to individuals. Simulation results are supplemented to illustrate the significant energy savings of the proposed strategies in providing reliable services.

This paper is organized as follows. The literature review of related work is given in Section 2. The system model and problem formulation are introduced and derived in

Section 3. The optimal cooperative computing scheme is presented and analyzed in Section 4. The energy efficient cooperation node selection strategies are proposed in Section 5. Simulation results are provided in Section 6. Finally, concluding remarks are given in Section 7.

2 Related Work

Cloud computing has been intensively investigated based on off-the-shelf cloud infrastructures, such as resources/traffic optimization of backhaul networks [16], services admission control [17] and scheduling [18], and pricing strategies of using commerce cloud services [19], [20], etc. However, existing cloud computing models are designed for traditional web applications, rather than future Internet applications running on various mobile and sensor nodes.

Due to the emerging development of mobile Internet, more recent works show great interests in dealing with mobile applications in the context of cloud computing. Some state-of-art literature [3], [21] in mobile cloud computing (MCC) reveal that the energy issue is one of the major challenges. The issue of energy efficiency in future computing has also been extended to the sensor cloud computing. Alamri et al. in [22] provide a comprehensive survey of sensor cloud architecture, approaches and applications. Perera et al. in [23] propose a middleware design for IoT and balance the computation and communication energy between sensor nodes and cloud servers. Yuriyama and Kushida in [24] propose the concept of virtual sensors by collecting different vertical application data into a single horizontal platform which can behave as a sensor node to reduce extra communication between networks. There are also sensor cloud applications in body sensor networks [25] and truck monitoring [26], etc. Although various sensor cloud schemes have been developed to increase bandwidth efficiency, the sensor nodes are usually assumed as data collecting points and there is lack of understanding of their processing capability and the potential benefits of being a computing node.

As a contrary, edge computing [27], [28] has been considered to provide computing, storage, and networking services between end devices and traditional Cloud Computing data centers, typically, but not exclusively located at the edge of network. A comparable concept has also been proposed by Cisco with the name of fog computing [29]. Since mobile sensor nodes now rival many PCs in terms of computational power [6], they have the opportunity to talk directly to one another when possible and handle much of their own computational tasks. Moreover, an emerging wave of sensor applications, requires mobility support and geo-distribution in addition to location awareness and low latency. In our preliminary study [30], we have already developed a prototype to connect an on-board diagnostics (OBD) sensor device to the cloud via smart phone, where the data analysis engine can be deployed in either smart phone or cloud, depending on the size of the processing data and service requirements. A similar concept of crowd computing [31] has also been proposed to bring together the strengths of crowdsourcing, automation and machine learning.

In this paper, our contribution is to further leverage the computation capability of sensor nodes into computing and consider a joint optimization of both computation and communication energy across mobile wireless sensor nodes to transfer and process sensor data and disseminate results to appropriate parties. To the best of our knowledge, this is the first work that considers sensor node as a service and realizes cooperative computing in MWSN.

3 System Model and Energy Formulation

In this section, we introduce the application model, its execution on sensor node and transmission over wireless channel. Specifically, the energy consumption of both computation and communication are formulated.

3.1 Application Model

We consider the cooperative computing in which each sensor node can execute lightweight applications with the help of peer sensor nodes. In order to characterize an application, a canonical model [32] that captures the essentials of a typical application is considered and can be abstracted into the following two parameters:

- *Input data size* L. The total number of data bits as the input of an application. Such input data can be partitioned and offloaded to a peer sensor node for remote processing and execution [33].
- *Application completion deadline* T. The maximum number of time slots that an application must be completed. t is the discrete time index ranging from $t = 1$ to $t = T$.

It is worth noting that an application is a program that performs a computation on an input file, such as calculating the number of violate data from a period of history record. Similar to the model applied in MapReduce [34], we consider that an application can be breakable into tasks which do not exhibit dependencies across partitions of its input. We assume that all sensor nodes are capable of executing a same application without need to transfer executable files for operation, thus only the input partitions are transmitted to other sensor nodes for parallel executions. Although there are cases that some tasks cannot be broken into smaller pieces and can only be executed on a single node due to the dependencies in its input, there are still concurrency benefits when many such tasks are executed in batches.

The energy consumption of an application is highly related to these two parameters. For example, with a large size of input data and stringent completion deadline, a sensor node may consume extensive energy. In the following, we denote such an application as $A(L, T)$ and use it to characterize the energy consumption of computation and communication, respectively.

3.2 Computation Energy Consumption

The energy consumption of computation is directly determined by the CPU workload of an application. According to [35], the workload can be measured by the number of required CPU cycles, which is related to the input data size and computation complexity, and is defined as

$$W = LX, \tag{1}$$

where W is the total number of required CPU cycles, L is the input data size and X is the computation algorithm. It is

noted that with a same size of input data, the required cycle demands often vary greatly, which depend on the nature of applications [36], e.g., applying an input data to calculate the average and factorial show distinct computation complexities. In the existing literature [36], [37], [38], X has been shown as a random variable and can be modeled by a Gamma distribution which is commonly used to model service times [39], and has been shown to work well in characterizing the distribution of CPU cycle demands [32], [37].

Although a number of factors consume CPU power, such as short circuit power and dynamic power, etc., the energy consumption is dominated by dynamic power which can be minimized by configuring the clock frequency of the chip via the dynamic voltage scaling (DVS) technology[2] [40]. As a result, the total energy consumption of computation is given by

$$E_c = \sum_{w=1}^{W} \epsilon_c(w) = \sum_{w=1}^{W} \kappa f_w^2 , \tag{2}$$

where ϵ_c is the computation energy per operation cycle, κ is the effective switched capacity determined by the chip architecture and f_w is the clock-frequency which is scheduled in the next CPU cycle given the number of w CPU cycles have been completed.

A careful reader may notice that the CPU can reduce its energy consumption by scheduling low clock frequency. However, as a practical implementation, the application has to meet a completion deadline. Without deadlines, there is no particular reason to complete any given task by a certain time. We use the soft deadline to characterize probabilistic performance, that is, the statistical CPU scheduling model [32] which assumes the application completion needs to meet its deadline with the probability p by allocating W_p CPU cycles. The parameter p is the application completing probability (ACP). In other words, the probability of an application requires no more than the allocated W_p should satisfy $F_W(W_p) = Pr[W \leq W_p] \geq p$. This soft real-time scheduling integrated with DVS has been shown its effectiveness in saving energy without substantially affecting application performance [36].

According to (1), since W is a linear function of X, we can obtain $W_p = LF_X^{-1}(p)$, where $F_X^{-1}(p)$ is the inverse cumulative distribution function of X. Therefore, the total energy consumption can be derived as

$$E_c = \kappa \sum_{w=1}^{W_p} F_W^c(w) f_w^2 , \tag{3}$$

where $F_W^c(w)$ is the complementary cumulative distribution function (CCDF) that the application has not completed after w CPU cycles. Since the Gamma distribution is exponentially tailed, the CCDF can be assumed as $F_W^c(w) \sim \mu e^{-\nu w}$ for some constants $\mu > 0$ and $\nu > 0$. It is noted that with $w \to \infty$, the probability goes to 0, which means it is unlikely that an application cannot be completed with a large number of CPU cycles.

Theorem 3.1 [32]. *For the optimal clock-frequency scheduling in each CPU cycle (f_w) and the deadline requirement ($\sum_{w=1}^{W_p} 1/f_w \leq T$), the minimum computation energy can be derived as*

$$E_c^* = \frac{\kappa}{T^2} \left\{ \sum_{w=1}^{W_p} [F_W^c(w)]^{1/3} \right\}^3 , \text{ and also has } E_c^* \sim L^3.$$

Simplifying the above result, we have the optimal computation energy as

$$E_c = \frac{KL^3}{T^2} . \tag{4}$$

where K is a constant factor determined by κ and p. The result tells that allocating a smaller size of input data or relaxing the deadline can achieve better computational energy efficiency, which means that a sensor node can prefer to execute delay tolerant applications or offload more tasks to peer node, in order to save its own computation energy.

3.3 Communication Energy Consumption

When a sensor node considers to offload its tasks to a peer node, the energy consumption is determined by the number of bits being transmitted over wireless channel.

The energy consumption of communication is determined by the current draw of the electrical circuits that implements the physical communication layer. In practice, it includes idle, transmit and receive modes. According to the specifications of IEEE 802.11n [41] or IEEE 802.15.4 [42], [43], the energy consumption of a wireless sensor node is dominated by the transmit or receive modes, and has their costs are approximately the same. So we consider the communication energy including both transmission and reception of processing tasks, and do not consider the small output results[3] from the node.

The communication cost is characterized by the empirical transmission energy model [45], and the required energy E_t to transmit L bits is governed by a convex monomial function[4]

$$E_t = \rho \frac{L^n}{g} , \tag{5}$$

where ρ denotes the energy coefficient, g denotes channel state and n denotes the order of monomial with value $1 \leq n \leq 5$. According to [45], the choice of n depends on the bit scheduler policy. With an increasing value of n, the scheduler more prefers to transmit equal number of bits at every time slot regardless of the channel state. However, in this paper, we consider the optimal case of $n = 1$ which is called one-shot policy [47], in which the transmission only depends

2. DVS is commonly used to save CPU energy by adjusting the speed based on required cycle demands. It exploits an important characteristic of complementary metal oxide semiconductor (CMOS)-based processors: When operation is at low voltage, the clock frequency scales as a linear function of the voltage supply. The energy consumed per cycle is proportional to the square of the voltage.

3. This is a reasonable assumption for sensor computing where most of sensor based applications come with simple results of warning or image detection indication [44], etc.

4. Although the monomial cost does not hold for operation at capacity in AWGN channel, there is a practical modulation scheme to well approximate by a monomial [46].

on the channel state and is completed in one time slot. There are several reasons for applying this scenario: First, for energy constrained sensor node, it may not be desirable to split a single data across multiple time slots because of extra energy consumed by a large overhead associated with each slot. Second, since we impose a deadline to complete an application, the transmission time should be relatively small compared to T, such that the time offset between local and remote executions can be negligible. Third, in MWSN, the transmission time over the air should be minimized to avoid channel fluctuation caused by node mobility. In order to ensure the optimal performance of the policy, the scheduler should be opportunistic, that is, to offload tasks to a peer node with good channel quality. The scheduling policy has been proved its effectiveness in combating channel fading [6].

4 Energy Optimization for Cooperative Computing

Our interest is to find an optimal partition to minimize the total energy consumption of processing an application given that a target completion deadline T is satisfied by using cooperative computing, and can be formulated as

$$\begin{aligned} \min_{l_l, l_r} \quad & \underbrace{E_c^l(l_l, t) + E_t(l_r, g_{l,r})}_{\text{local energy cost}} + \underbrace{E_r(l_r, g_{r,l}) + E_c^r(l_r, t)}_{\text{remote energy cost}} \\ s.t. \quad & l_l + l_r = L \\ & t \le T\,. \end{aligned} \tag{6}$$

- E_c^l and E_c^r denote the local and remote computation energy consumption, and E_t and E_r denote transmission and reception energy consumption, respectively.
- l_l and l_r are partitioned input data size for local and remote processing. A symmetric channel is assumed between local and remote sensor nodes and has channel gain $g_{l,r} = g_{r,l}$. A completion deadline T is considered to ensure quality-as-service.

Theorem 4.1. *The optimal input data partition to minimize the total energy consumption of processing an application $A(L, T)$ in MWSN, is given by*

$$l_l^* = \frac{L}{2} + \frac{\beta}{6\alpha L}, \quad l_r^* = \frac{L}{2} - \frac{\beta}{6\alpha L}\,. \tag{7}$$

The corresponding minimum total energy consumption is

$$E_{total}(l_l^*, l_r^*) = \frac{\alpha L^3}{4} + \frac{\beta L}{2} - \frac{\beta^2}{12\alpha L}\,. \tag{8}$$

where $\alpha = \frac{K}{T^2}$, $\beta = \frac{2\rho}{g_{l,r}}$, K denotes the computation coefficient, ρ denotes communication coefficient of wireless channel and $g_{l,r}$ is the channel gain.

Proof. See Appendix A. □

In general, we find that the minimum total energy consumption can be achieved by optimally partitioning, offloading and executing the input data via cooperative computing, which can be determined by the application profile, hardware configurations of sensor nodes and wireless channel conditions. In the following, we further investigate the individual behaviors of the optimal partition and analyze how the system parameters affect the overall performance.

Proposition 4.2. *The size difference of the optimal input data between local and remote executions is*

$$diff(L, T, K, \rho, g_{l,r}) = \frac{2\rho T^2}{3 g_{l,r} K L}\,. \tag{9}$$

Proof. The result follows (7) and has $diff = \frac{\beta}{3\alpha L}$. Using $\alpha = \frac{K}{T^2}$ and $\beta = \frac{2\rho}{g_{l,r}}$ into the result leads to (9). □

In essence, we can observe that the optimal partition highly depends on system parameters. Specifically, *the local execution is preferable* when (9) tends to increase (i.e., small data size L, long completion deadline T, high transmission cost ρ, low computation cost K or high channel loss $g_{l,r}$). Otherwise, *the remote execution is preferable*.

Result 4.3. By defining the application processing speed as $\upsilon = \frac{L}{T}$, we have the equivalent energy optimal computing rules

$$\begin{cases} \text{Local computing}, & \text{if} \quad 0 < \upsilon \le \sqrt{\frac{2\rho}{3Kg_{l,r}}}, \\ \text{Cooperative computing}, & \text{if} \quad \sqrt{\frac{2\rho}{3Kg_{l,r}}} < \upsilon \le \sqrt{\frac{2\rho}{\sqrt{3}Kg_{l,r}}}, \\ \text{Cloud computing}, & \text{if} \quad \sqrt{\frac{2\rho}{\sqrt{3}Kg_{l,r}}} < \upsilon. \end{cases} \tag{10}$$

Proof. According to (7), since $\frac{L}{2} \le l_l^* \le L$, we should have $0 \le \frac{\beta}{6\alpha L} < \frac{L}{2}$ for data offload. Thus, the lower bound of application processing speed can be achieved as

$$\frac{L}{T} > \sqrt{\frac{2\rho}{3Kg_{l,r}}}, \tag{11}$$

when $\frac{\beta}{6\alpha L} \ge \frac{L}{2}$, only the local computing is applied. Furthermore, considering the cloud computing case where sensor nodes only collect and transmit input data to an enterprise cloud server via gateway for execution, we obtain the upper bound condition of using cooperative computing from (5) and (8) as

$$E_{total}(l_l^*, l_r^*) \le E_t \Rightarrow \frac{\alpha L^3}{4} \le \frac{\beta^2}{12\alpha L} \Rightarrow \frac{L}{T} \le \sqrt{\frac{2\rho}{\sqrt{3}Kg_{l,r}}}. \tag{12}$$

□

We observe that when the application prefers cooperative computing, it can always achieve better energy efficiency than the local computing case. Moreover, the cloud computing is only applied when high processing speed is required.

Proposition 4.4. *For cooperative computing, the bound performance of computation to communication energy ratio of local and remote nodes are*

$$\frac{E_c^l}{E_t} \ge \frac{Kg_{l,r}}{4\rho}\upsilon^2 \ge \frac{E_c^r}{E_r}\,. \tag{13}$$

Proof. See Appendix B. □

The result tells that the proposed optimal solution can best achieve the local computation energy and have the minimal energy ratio. Assuming the application processing speed can be closed to its lower bound in (11), the local computation energy can be achieved as close as $1/6$ of transmission energy, which is promising to show the advantage of using cooperative computing in MWSN. Such ratio can be higher, depends on the preference of application processing strategy and system parameters. Moreover, the cooperative computing can also help reduce the remote computation energy. Especially, with fewer offloading bits, the remote computation energy decreases at a faster pace than the communication energy. The upper bounded is govern by the system parameters and maximum number of bits can be offloaded. The result is useful when selfish nature is imposed to individuals, because reducing the energy consumption for helping nodes can largely improve their willingness of cooperation.

In addition to the individual performance, we also provide an analytical result to characterize the average performance of the optimal solution. We assume that sensor candidates are randomly located in space according to a Poisson point process with density λ. An application initial node (IN) will choose the best cooperation node (CN) to achieve the minimum total energy among all available candidates. A network with a higher density of sensor nodes can have better choices to select CN.

We consider a simple channel model where $g_{i,j}$ between the nodes i and j is modelled as $g_{i,j} = 1/d_{i,j}^a$, where $d_{i,j}$ is the distance between the nodes i and j, a is the path-loss exponent and usually characterized as an integer value $a \geq 2$. Given the identical system parameters (i.e., K and ρ), the selected CN distance to achieve the minimum E_{total} will be as close as possible to the IN. We let r^* be a random variable of the selected CN distance to the IN, and r denote the distance between the closest CN and the IN. The probability distribution function of r is given by

$$\begin{aligned}\Pr[r^* < r] &= 1 - \Pr[r^* \geq r] \\ &= 1 - \Pr[N_r = 0] = 1 - e^{-\lambda\pi r^2},\end{aligned} \tag{14}$$

where N_r is the number of CN within distance r from the IN. The probability density function (pdf) of the selected CN distance is

$$f(r) = 2\lambda\pi r e^{-\lambda\pi r^2}, \quad r \geq 0. \tag{15}$$

According to (8) and (15), the expected value of the optimal energy consumption is

$$\begin{aligned}\mathrm{E}[E_{total}] &= 2\lambda\pi \int_0^\infty \left(\frac{\alpha L^3}{4} + \rho L r^a - \frac{\rho^2 r^{2a}}{3\alpha L}\right) r e^{-\lambda\pi r^2} dr \\ &= 2\lambda\pi \left[\int_0^\infty \frac{\alpha L^3}{4} r e^{-\lambda\pi r^2} dr + \int_0^\infty \rho L r^{a+1} e^{-\lambda\pi r^2} dr \right. \\ &\quad \left. - \int_0^\infty \frac{\rho^2 r^{2a+1}}{3\alpha L} e^{-\lambda\pi r^2} dr\right] \\ &= \begin{cases} \frac{\alpha L^3}{4} + \frac{\rho L a!!}{2^{\frac{a+1}{2}} (\lambda\pi)^{\frac{a-1}{2}}} \sqrt{\frac{1}{\lambda}} - \frac{\rho^2 a!}{3\alpha L (\lambda\pi)^a}, & \text{if } a \text{ is odd}, \\ \frac{\alpha L^3}{4} + \frac{\rho L (\frac{a}{2})!}{(\lambda\pi)^{\frac{a}{2}}} - \frac{\rho^2 a!}{3\alpha L (\lambda\pi)^a}, & \text{if } a \text{ is even}, \end{cases}\end{aligned} \tag{16}$$

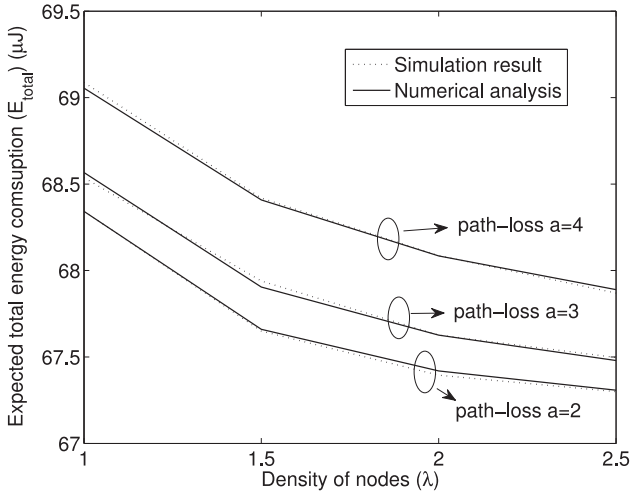


Fig. 2. Comparison of optimal energy consumption between simulation and numerical analysis: $L = 1,024\,\text{bits}$, $T = 20\,\text{ms}$, $\rho = 0.006$ and $K = 10^{-10}$.

where !! is the double factorial and have $a!! = \prod_{i=1}^{(a+1)/2} (2i-1)$. Fig. 2 illustrates the optimal energy comparison of the simulation result and numerical analysis (16) under different path-loss exponents. We can learn from the result is that with an increasing value of λ, there is a better chance to select a well positioned sensor node to improve the energy efficiency of the proposed solution. Moreover, with a higher loss of the channel, the optimal solution tends to local computing, which increases the energy consumption. In the following, we focus on a multi-node scenario and design energy efficient CN selection strategies.

5 Energy Efficient Cooperation Node Selection Strategies in MWSN

In this section, we consider a more general network setting where multiple nodes co-exist and cooperate with each other by acting as CN in cooperative computing.

We are interested in finding a strategy that each application IN determines which node to select as the CN for the maximal energy efficiency in the multi-node environment. It is noted that CN selections affect the overall energy consumption, since the optimal energy cost depends upon the application requirement, sensor hardware configuration and channel condition of the selected CN. In the following, we summarize the system parameters and assumptions for the multi-node environment.

- The network consists of a set of nodes $N = \{1, ..., n\}$, where each node $i \in N$ processes a number of application tasks over time. For simplicity, we assume all tasks have the same application profile $A(L, T)$, though it is straight-forward to derive CN selection rules in a more general setup. We also assume that time is divided into discrete time slots.
- One sensor node can only execute one application task at a given time slot and we do not consider multi-tasking scenarios.
- We denote by $E_{i,j}^{\mathrm{IN}}(t)$ the energy cost of a node i given that the node j is chosen as the CN at time t for cooperative computing. Meanwhile, the energy cost of the CN j is denoted as $E_{i,j}^{\mathrm{CN}}(t)$. We assume that the

node i and j follow the optimal partition given by (7) for each executions, and both computation and communication energy can be estimated by (4) and (5), respectively.

- When node i uses local computing at time t, we denote its energy cost as $E_i^{\mathrm{L}}(t)$.
- Energy consumption of a node $E_i(t_1 : t_2)$ during a time interval $[t_1 : t_2]$ is the sum of node i's execution energy either as IN or CN, and communication energy either transmission or reception over all $t \in [t_1, t_2]$ (we assume a node consumes zero-energy at t if it is neither an IN or a CN at t).
- We denote $\mathcal{R}_i(t)$ as the set of the nodes (except node i) which can achieve energy saving compared to the local computing, for processing node i's application at time t, i.e., $\mathcal{R}_i(t) = \{j \in N - \{i\} | E_{i,j}^{\mathrm{IN}}(t) + E_{i,j}^{\mathrm{CN}}(t) < E_i^{\mathrm{L}}(t)\}$.

5.1 Minimum Total Energy CN Selection

Our first CN-selection strategy makes use of the result in previous section in a straightforward manner:

Min-Total-Energy-Strategy. A CN is selected for node i at time t such that

$$CN_i(t) = \arg \min_{j \in \mathcal{R}_i(t)} \{E_{i,j}^{\mathrm{IN}}(t) + E_{i,j}^{\mathrm{CN}}(t)\}.$$

In other words, for each processing task from node i, a CN j is selected, which minimizes $E_{i,j}^{\mathrm{IN}}(t) + E_{i,j}^{\mathrm{CN}}(t)$ among those in $\mathcal{R}_i(t)$. If $\mathcal{R}_i(t) = \emptyset$, $CN_i(t) = null$.

Result 5.1. For a given time interval of $[t_1, t_2]$, the total energy consumption of MWSN $\sum_{i \in N} E_i[t_1, t_2]$ is minimized if each i is assigned a CN at each time by the Min-Total-Energy-Strategy, i.e., $CN_i(t) = \arg\min_{j \in \mathcal{R}_i(t)} E_{i,j}^{\mathrm{IN}}(t) + E_{i,j}^{\mathrm{CN}}(t)$.

Proof. Since we only consider the total energy consumption, the whole cooperative computing process can be scheduled into several rounds and each node can only process no more than one application task in each round. Since any assignment CN_i is injective in each round, for any two nodes i and k, $S_i \cap S_k = \emptyset$, and $\cup_{i \in N} S_i \subseteq N$, where S_i is a set of IN whose CN is i. Therefore, the total energy consumption in each round $\sum_{i \in N} E_i = \sum_{i \in N} (E_{i,j}^{\mathrm{IN}} + \sum_{j \in S_i} E_{j,i}^{\mathrm{CN}})$ can be re-written as $\sum_{i \in N} E_{i,j}^{\mathrm{IN}} + \sum_{i \in N} \sum_{j \in S_i} E_{j,i}^{\mathrm{CN}} = \sum_{i \in N} E_{i,CN_i}^{\mathrm{IN}} + \sum_{j \in N} E_{j,CN_j}^{\mathrm{CN}} = \sum_{i \in N} (E_{i,CN_i}^{\mathrm{IN}} + E_{i,CN_i}^{\mathrm{CN}})$, which is minimized if each individual term is minimum. □

It is worth noting that this result is straightforward and obtainable from the optimal partition in (7), since the CN selection is based only on the energy consumption of itself and other potential CN nodes for the upcoming task at each t, but not on the past energy consumptions of itself or other nodes. However, it is clear that, though simple, the Min-Total-Energy-Strategy is optimal in the sense that it minimizes the total energy consumption of MWSN.

However, from the individual's perspective, we may argue that the CN selection can lead to the situation that some nodes end up with higher energy consumption than the case when all nodes employ local computing. This is especially true if some unfortunate nodes are heavily selected as CN and hence consume more energy in cooperation than that saved from its own processing as an IN. In the following, we consider how to handle such unfairness in cooperative computing.

5.2 Fairness CN Selection

Our strategy is to let each node act as a CN only when it has saved more energy than that it has lost from cooperative computing in the past. Thus, we define the following functions to indicate the availability of each node.

5.2.1 Utility Function

To represent how much energy saving of cooperative computing can yield in comparison to local computing, we introduce the concept of "utility" of nodes.

The *utility function*, $u_i(t)$, of node i at time t is defined as

$$u_i(t) = \begin{cases} E_i^{\mathrm{L}}(t) - E_{i,j}^{\mathrm{IN}}(t), & \text{if } \exists j \text{ s.t., } j = CN_i(t), \\ -E_{j,i}^{\mathrm{CN}}(t), & \text{if } i = CN_j(t) \text{ for some IN } j, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

The above function represents how much energy a node i locally saves (or loses) compared to local computing at time t, where $E_i^{\mathrm{L}}(t) - E_{i,j}^{\mathrm{IN}}(t)$ denotes the energy saved from i's cooperative computing using a CN j at time t, and $-E_{j,i}^{\mathrm{CN}}(t)$ denotes the energy cost by node i as a CN for some other node j at time t. In all other cases (if i is not active either as an IN or a CN, or if i uses local computing), the utility is 0. The initial $u_i(t)$ can be any arbitrary value, but for simplicity, we assume $u_i(t) = 0$ for all $i \in N$. Then the cumulative utility over a time interval $[t_1 : t_2]$ is defined as $u_i(t_1 : t_2) = \sum_{\tau=t_1}^{t_2} u_i(\tau)$, which is the overall energy savings of a node during the time interval.

5.2.2 Cooperation Index

$$C_i(t) = \begin{cases} 1, & \text{if } u_i(0 : t-1) \geq 0, \\ 0, & \text{if } u_i(0 : t-1) < 0. \end{cases} \tag{18}$$

This value[5] is used to decide whether node i can act as a CN for other nodes (when $C_i(t) = 1$, i.e., in "cooperation" mode) or i should not be selected as a CN for any other node (when $C_i(t) = 0$). It is maintained for each node i and updated at each time t.

Adaptive-Positive-Utility-Strategy. A CN is selected for node i at time t such that

$$CN_i(t) = \arg \min_{j \in \mathcal{R}_i(t), C_j(t)=1} \{E_{i,j}^{\mathrm{IN}}(t) + E_{i,j}^{\mathrm{CN}}(t)\}. \tag{19}$$

In other words, a CN j is selected for the i's cooperative computing at time t that minimizes $E_{i,j}^{\mathrm{IN}}(t) + E_{i,j}^{\mathrm{CN}}(t)$ among the nodes whose cumulative utilities are positive or zero.

5. We set $C_i(0) = 1$ in order to enable the initial cooperation condition when all nodes's utilities are zero. If $C_i(0) = 0$ for all i, no node would cooperate to other nodes.

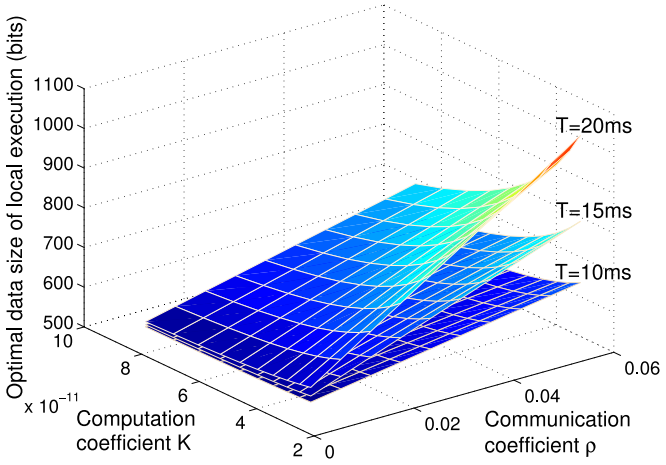


Fig. 3. The relations between the optimal data size of local execution and system coefficients K and ρ.

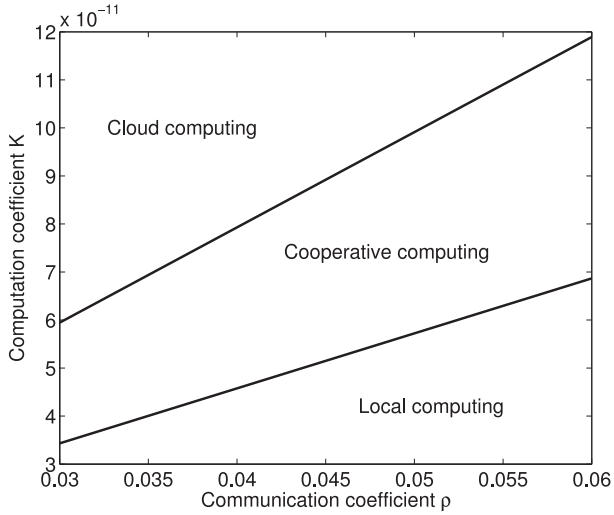


Fig. 4. An illustration of energy optimal computing rules.

Thus, a node whose cumulative utility is negative will cease to act as a CN, and will be potentially available as a CN when its utility becomes positive. Note that the min-total-energy-strategy can be seen as a special case of the adaptive-positive-utility-strategy with $C_i(t) = 1$ for all i and for all t.

Giving that some nodes may benefit more from better cooperation opportunities (i.e. larger utilities) than the others due to difference in the amount of tasks and to potentially unfair channel conditions, we can generalize the strategy to bring the balance (or "fairness") of the amount of utilities that individual nodes collect.

5.2.3 Fairness Factor

How much importance will be given to the fairness term, reflecting the utility and how much to the energy consumption term, depends on how fast the function $w(u)$ decays as the utility value u increases. Motivated by [48] that the well-established power-law method can measure the fairness and inequality of network performance, we employ a power-law function

$$w(u) = u^{-k}, \tag{20}$$

where parameter k is a positive constant and can be used to tradeoff fairness in energy consumption. In our simulation study, we find that $w(u) = u^{-5}$ strikes a good balance. The principal implication of this power-law relationship is that the CN selection is far from random, i.e., a node with a larger utility (smaller weight) will have a higher chance to be selected as a CN, so the $w(u)$ obtained from the utility function can help us improve the energy performance of the system.

Considering all above strategies, we propose a Weighted-Fairness-Strategy to bring the fairness of the amount of utilities that individual nodes collect.

Weighted-Fairness-Strategy. A CN is selected for node i at time t such that

$$CN_i(t) = \arg \min_{\substack{j \in \mathcal{R}_i(t), \\ C_j(t)=1}} \{w(u_j(0:t-1))(E_{i,j}^{\mathrm{IN}}(t) + E_{i,j}^{\mathrm{CN}}(t))\}, \tag{21}$$

where $w(u) = u^{-5}$ is a non-increasing function of the utility value u. The constraint $C_j(t) = 1$ ensures that a node whose cumulative utility is negative will cease to act as a CN, and will be potentially available as a CN when its payoff becomes positive. Here, along with the energy consumption factor $(E_{i,j}^{\mathrm{IN}}(t) + E_{i,j}^{\mathrm{CN}}(t))$, the weight function $w(u_j(0:t-1))$ is introduced in the CN selection strategy, such that the nodes with larger utilities (i.e., smaller weight) will have a higher chance to be selected as a CN for each task execution. Additionally, among CNs which have the same total energy consumption, preference will be given to the ones with higher cumulative utilities.

6 Simulation Results

In this section, we evaluate performance of the proposed optimal solution via numerical and simulation results obtained using MATLAB. To be consistent with the real energy measurements [35], we set the computation coefficient in the order of 10^{-11}, the communication coefficient in the order of 10^{-2}, a time slot $t = 2\,\mathrm{ms}$ and channel gain $0 < g < 1$.

6.1 Performance of the Optimal Cooperative Computing

Fig. 3 shows the local data size partitioned by the proposed optimal solution. The input data size is assumed as $L = 1024\,\mathrm{bits}$ and channel gain is $g_{l,r} = 0.5$. It is clear that the optimal partition is significantly affected by the system coefficients. With better computation efficiency (smaller K) and higher communication cost (larger ρ), the optimal partition tends to allocate more processing task locally. Moreover, with a relaxed completion deadline (large T), the local execution is more preferable to save energy by reducing processing speed.

Fig. 4 gives an illustration of the energy optimal computing rules for $L = 1,024\,\mathrm{bits}$, $T = 30\,\mathrm{ms}$ and $g_{l,r} = 0.5$. With the application profile and system coefficients, we can quickly decide the best strategy to process an application.

Fig. 5 shows the total energy consumption of the optimal solution and compare it with that of local computing. By setting $K = 10^{-10}$, $\rho = 0.006$ and $T = 20\,\mathrm{ms}$, we observe that under the same communication coefficient, the energy performance improves with better channel quality. Even with severe channel quality and high communication cost ($g_{l,r} = 0.1$, $\rho = 0.01$), the performance of the proposed solution is closed to the local computing when the application

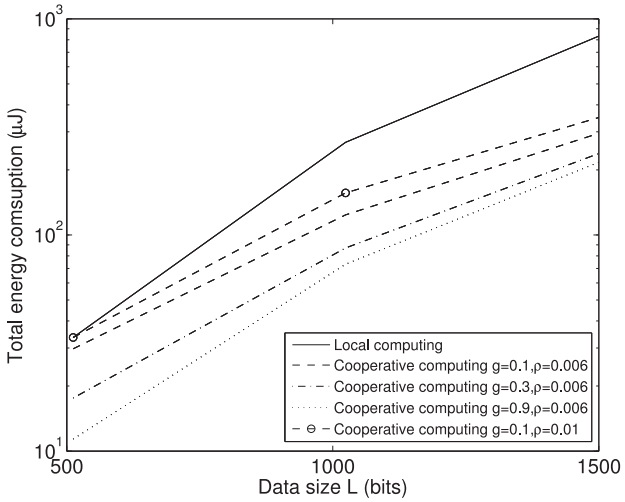


Fig. 5. Total energy consumption versus input data size.

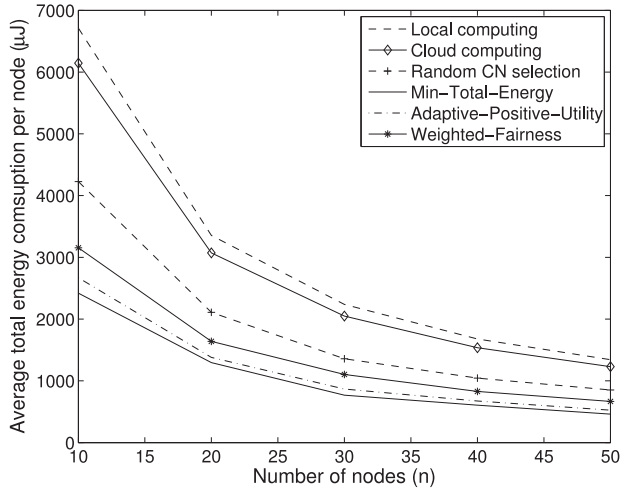


Fig. 7. Average total energy consumption per node for $L = 1,024\,\text{bits}$.

requirement is not stringent (small L, large T). As the input data size increases, the cooperative computing can ensure optimal with better energy efficiency than the local computing. Given the worst channel scenario with $g_{l,r} = 0.1$, an average of 63 percent of energy can still be saved by using the proposed cooperative computing.

Fig. 6 shows the energy comparisons in terms of completion deadline. We set $K = 10^{-10}$, $\rho = 0.01$ and $L = 512\,\text{bits}$. With a longer completion time, the optimal solution can reduce energy consumption by adjusting the process speed at a lower pace. Even for the worst channel scenario, the optimal solution can still achieve better performance and close to the local computing when the delay tolerance is high. Moreover, consider the cloud computing where sensor nodes only collect and transmit data to the enterprise cloud server, the proposed cooperative computing can still achieve energy efficiency on handling non-emergent applications.

6.2 Performance of the CN Selections for Cooperative Computing

We consider the size of MWSN is N (varied between 10 and 50 sensor nodes). Throughout the simulation, we set the input data size $L = 1024\,\text{bits}$ and completion deadline $T = 40\,\text{ms}$. The channel gain between two nodes is randomly selected between $0 < g < 1$ and will be changed over the time, but kept constant during one offloading process. A total of 1,000 application tasks are executed within the network, and at each time t, an application is executed by a random IN and a selected CN. The initial utility value of every node $u_i(0)$ is set 0.

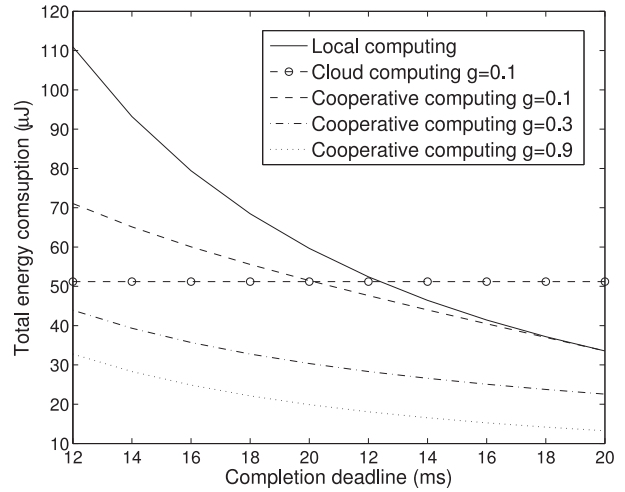


Fig. 6. Total energy consumption versus completion deadline.

Fig. 7 shows the average energy consumption per node. The proposed CN selection strategies outperform the local computing and cloud computing. However, since the min-total-energy strategy is the optimal solution in this case, the weighted-farness strategy performs a bit worse (this is compensated by fairness results). Furthermore, as the number of nodes increases, the average energy consumption of the proposed CN strategies decreases, this is because it is easier to find a CN with better channel quality and thus save more energy. As a comparison, Fig. 8 shows the average performance for $L = 512\,\text{bits}$ with the same deadline $T = 40\,\text{ms}$. It is clear that with less stringent application requirement, it is preferable to employ the proposed solutions. The cloud computing strategy which is commonly used in today's sensor cloud solution is not always optimal in dealing with non-emergent applications.

Fig. 9 shows the fairness in terms of how much energy is saved for individual nodes using the proposed CN selection strategies, where the y-axis represents Jain's fairness index of nodes' cumulative utilities.[6] It is clear that the weighted-fairness strategy achieves the best fairness compared with other strategies. Moreover, the index curve shows a non-decreasing tendency toward increased total number of nodes, which is contradictory to the min-total-energy strategy where an increasing number of good quality nodes could be extensively used for cooperation. As another example to highlight the fairness, we show in Fig. 10 the energy consumption saving of individual nodes at the end of simulation in five-node network. It is clear that the weighted-fairness strategy achieves the best fairness, whereas the Min-Total-Energy results in negative utility for a node.

6. Jain's fairness index is defined by $(\sum U_i)^2/(N\sum U_i^2)$. The result ranges from $\frac{1}{N}$ (worst case) to 1 (best case). The larger the index is, the better fairness that we can achieve.

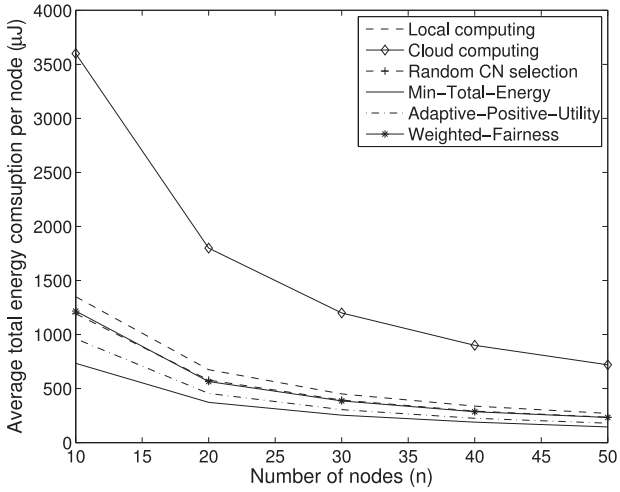


Fig. 8. Average total energy consumption per node for $L = 512\,\text{bits}$.

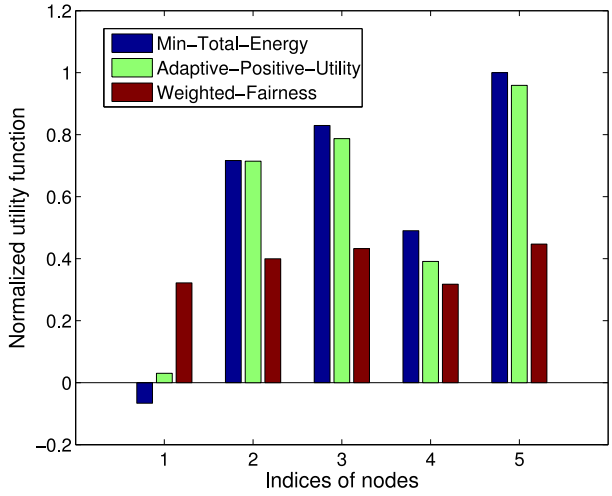


Fig. 10. Normalized utility function per node.

We provide additional measurement of the impact of each node's "willingness" to cooperate when its utility is zero when the weighted-fairness strategy is used. We therefore change the rule (18) for a subset of nodes, and divide the nodes into two groups: $\mathcal{U} = \{i \mid C_i(t) = 1 \text{ if } u_i(t) = 0\}$ ('Unselfish node'), and $\mathcal{S} = \{i \mid C_i(t) = 0 \text{ if } u_i(t) = 0\}$ ('Selfish node'); the rule remains the same as (5.2) for both group when $u_i(t) \neq 0$. In Fig. 11, we show the proportion of the nodes with positive utility in the y-axis as the time progresses in x-axis. The unselfish cooperation represents a total number of 100 unselfish nodes, whereas the selfish cooperation represents only one node cooperates initially to others out of 99. The result tells that the proposed fairness strategy can ensure the proportion of nodes with positive utility converges to 1, even for the extreme selfish scenario. Moreover, convergence speed is faster with a larger processing task, which indicates the proposed method is robust to cope with different application requirements and thus can ensure energy efficiency on individual nodes.

7 Conclusion and Future Work

We have shown that it is advantageous to employ cooperative computing to process application tasks, which can significantly reduce the total energy consumption while maintaining a given level of completion requirement. Specifically, we proposed a joint optimization problem of computation and communication costs as a whole, to optimally partition, offload and execute tasks between sensor nodes to boost the energy efficiency of edge computing. By implementing the proposed optimal solution into a network with multiple mobile wireless sensor nodes, the proposed cooperation node selection strategies can serve as an effective tool to achieve a desirable tradeoff between fairness and energy consumption at each node. The resulting ideas have the potential to have a broad impact across a range of areas, including Internet-of-Things, Machine-to-Machine and mobile cloud computing, etc.

In the future work, the following research issues will be considered: 1) *Cross-layer optimization of sensor cloud networks*: we will further incorporate characteristic of sensor networks into considerations, such as multi-hop transmission and resource constrains. Specifically, we will consider a practical application scenario where the routing protocol for low power and lossy network (RPL) [44] is used for smart grid application, and obtain an optimal cooperative solution to maximize the network lifetime. 2) *Multi-node cooperation*: So far we have focused on the single IN-CN pair

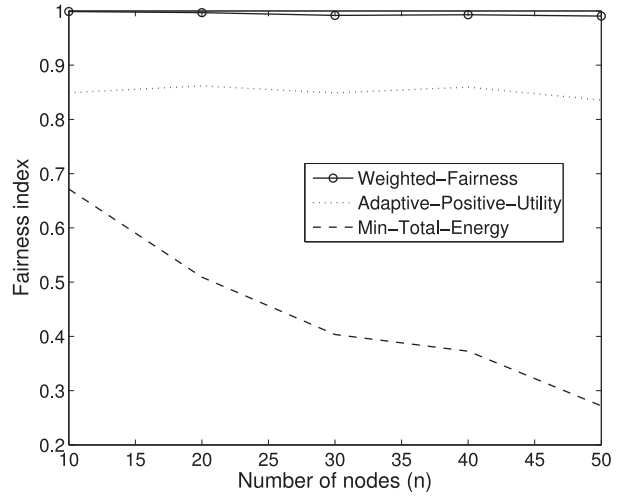


Fig. 9. Jain's fairness on utility function.

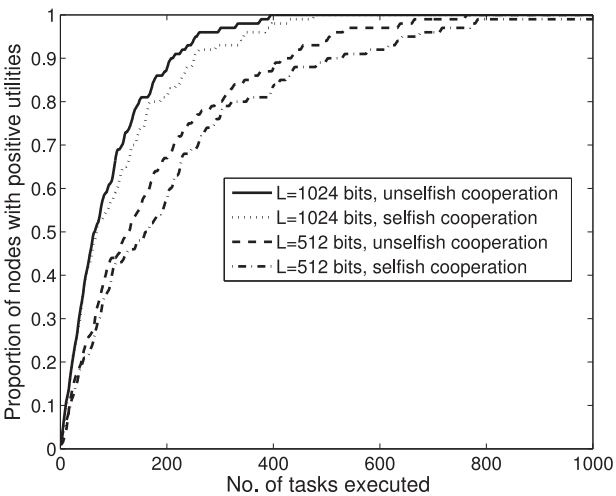


Fig. 11. Proportion of nodes with positive utility.

case. It would be also interesting to consider the multi-CN case where more than one cooperation nodes can be selected for sensor cloud computing. The challenges will be the new characteristic of communication energy model, since multiple subtasks need to be distributed independently to a number of CN along multiple time slots. Moreover, the trade-off between the overall energy performance and number of CN should be justified.

APPENDIX A
PROOF OF THEOREM 4.1

We use the Lagrange multiplier method to solve the optimization problem. According to (4) and (5), the optimization problem in (6) can be written as

$$\min_{l_l,l_r} \frac{Kl_l^3}{t^2} + \rho\frac{l_r^n}{g_{l,r}} + \rho\frac{l_r^n}{g_{r,l}} + \frac{Kl_r^3}{t^2}, \quad s.t. \quad l_l + l_r = L, t \le T\,. \tag{22}$$

In order to simplify the notation, we use $g_{l,r}$ to denote $g_{r,l}$ because of the symmetric channel assumption, and $n = 1$. According to the Kuhn-Tucker condition (p. 244: KKT conditions for convex problems [49]), the inequality constraint in (22) can be converted to the equality constraint and have the convex function

$$\ell(l_l, l_r, \lambda) = \frac{K}{T^2} l_l^3 + \rho\frac{l_r}{g_{l,r}} + \rho\frac{l_r}{g_{l,r}} + \frac{K}{T^2} l_r^3 + \lambda(l_l + l_r - L). \tag{23}$$

Let $\alpha = \frac{K}{T^2}$ and $\beta = \frac{2\rho}{g_{l,r}}$, we can derive the optimal partition which must satisfy the following conditions:

$$\frac{\partial \ell(l_l, l_r, \lambda)}{\partial l_l} = 3\alpha l_l^2 + \lambda, \tag{24}$$

$$\frac{\partial \ell(l_l, l_r, \lambda)}{\partial l_r} = 3\alpha l_r^2 + \beta + \lambda\,, \tag{25}$$

Then we obtain

$$l_l^2 = \frac{-\lambda}{3\alpha}, \quad l_r^2 = \frac{-\lambda - \beta}{3\alpha}\,. \tag{26}$$

Since $l_l + l_r = L$, we have

$$\lambda = -\frac{3\alpha L^2}{4} - \frac{\beta^2}{12\alpha L^2} - \frac{\beta}{2}\,. \tag{27}$$

Substituting (27) into (26), we obtain the unique optimal solution.

The optimal result (8) can thus be directly obtained by summing (4) and (5) for both local and remote executions with optimal partition (26).

APPENDIX B
PROOF OF PROPOSITION 4.5

1) *For local execution.* According to (4) and (5), we obtain $\frac{E_c^l}{E_t} = \frac{Kl_l^3}{T^2} \cdot \frac{g_{l,r}}{\rho l_r}$. Since $l_r \le l_l$, we have

$$\frac{E_c^l}{E_t} \ge \frac{Kl_l^3}{T^2} \cdot \frac{g_{l,r}}{\rho l_l} \Rightarrow \frac{E_c^l}{E_t} \ge \frac{Kg_{l,r}}{\rho} \cdot \left(\frac{l_l}{T}\right)^2. \tag{28}$$

Because $\frac{L}{2} \le l_l^* \le L$, we can obtain the lower bound performance of local execution as

$$\frac{E_c^l}{E_t} \ge \frac{Kg_{l,r}}{\rho} \cdot \left(\frac{L}{2T}\right)^2. \tag{29}$$

2) *For remote execution.* Similarly we have $\frac{E_c^r}{E_r} = \frac{Kg_{l,r}}{\rho} \cdot \left(\frac{l_r}{T}\right)^2$. Since $0 \le l_r^* \le \frac{L}{2}$, we can obtain the upper bound performance of remote execution

$$\frac{E_c^r}{E_r} \le \frac{Kg_{l,r}}{\rho} \cdot \left(\frac{L}{2T}\right)^2. \tag{30}$$

ACKNOWLEDGMENTS

This work was supported by the Start-Up Fund from the University of Sussex, UK, and in part by the Canadian Natural Sciences and Engineering Research (NSERC), the NSERC DIVA Strategic Research Network.

REFERENCES

[1] M. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for IT and scientific research," *IEEE Internet Comput.*, vol. 13, no. 5, pp. 10–13, Sep. 2009.

[2] M. Chen, H. Jin, Y. Wen, and V. Leung, "Enabling technologies for future data center networking: A primer," *IEEE Netw.*, vol. 27, no. 4, pp. 8–15, Jul. 2013.

[3] Y. Xu and S. Mao, "A survey of mobile cloud computing for rich media applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 46–53, Jun. 2013.

[4] B. Furht and A. Escalante, *Handbook of Cloud Computing*. New York, NY, USA: Springer, 2010.

[5] Edge computing. [Online]. Available: http://en.wikipedia.org/wiki/Edge_computing, 2015.

[6] M. Y. Arslan, I. Singh, S. Singh, H. V. Madhyastha, K. Sundaresan, and S. V. Krishnamurthy, "Computing while charging: Building a distributed computing infrastructure using smartphones," in *Proc. 8th Int. Conf. Emerging Netw. Exp. Technol.*, 2012, pp. 193–204.

[7] NVIDIA says Tegra 3 is a PC-class CPU. [Online]. Available: http://engt.co/srvibU

[8] A. Chandra, Y. Lee, B. M. Kim, S. Y. Maeng, S. H. Park, and S. R. Lee, "Review on sensor cloud and its integration with Arduino based sensor network," in *Proc. Int. Conf. IT Convergence Security*, Dec. 2013, pp. 1–4.

[9] S. Gu, Y. Yue, C. Maple, C. Wu, and B. Liu, "Challenges in mobile localisation in wireless sensor networks for disaster scenarios," in *Proc. 19th Int. Conf. Autom. Comput.*, Sep. 2013, pp. 1–6.

[10] T. Yamanoue, K. Oda, and K. Shimozono, "A M2M system using Arduino, Android and Wiki software," in *Proc. IIAI Int. Conf. Adv. Appl. Informat.*, Sep. 2012, pp. 123–128.

[11] Y. Zhang, L. Sun, H. Song, and X. Cao, "Ubiquitous WSN for healthcare: Recent advances and future prospects," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 311–318, Aug. 2014.

[12] E. Hossain, G. Chow, V. C. Leung, R. D. McLeod, J. Mišić, V. W. Wong, and O. Yang, "Vehicular telematics over heterogeneous wireless networks: A survey," *Comput. Commun.*, vol. 33, no. 7, pp. 775–793, 2010.

[13] G. Strazdins, A. Elsts, K. Nesenbergs, and L. Selavo, "Wireless sensor network operating system design rules based on real-world deployment survey," *J. Sens. Actuator Netw.*, vol. 2, no. 3, pp. 509–556, 2013.

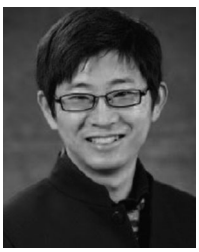
[14] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V. Leung, "Lightweight management of resource constrained sensor devices in internet-of-things," *IEEE Internet Things J.*, vol. PP, no. 99, p.1, Apr. 2015.

[15] Z. Sheng, C. Mahapatra, C. Zhu, and V. Leung, "Recent advances in industrial wireless sensor networks toward efficient management in IoT," *IEEE Access*, vol. 3, pp. 622–637, 2015.
[16] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1060–1071, Jun. 2013.
[17] H. Xu and B. Li, "A general and practical datacenter selection framework for cloud services," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, 2012.
[18] T. He, S. Chen, H. Kim, L. Tong, and K.-W. Lee, "Scheduling parallel tasks onto opportunistically available cloud resources," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jun. 2012, pp. 180–187.
[19] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2, pp. 158–171, Jul. 2013.
[20] V. Di Valerio, V. Cardellini, and F. Lo Presti, "Optimal pricing and service provisioning strategies in cloud systems: A Stackelberg game approach," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jun. 2013, pp. 115–122.
[21] C.-A. Chen, M. Won, R. Stoleru, and G. Xie, "Energy-efficient fault-tolerant data storage amp; processing in mobile cloud," *IEEE Trans. on Cloud Computing*, vol. 3, no. 1, pp. 28–41, Jan.–Mar. 2015.
[22] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, "A survey on sensor-cloud: Architecture, applications, and approaches," *Int. J. Distrib. Sensor Netw.*, vol. 2013, p. 917923, 2013.
[23] C. Perera, P. P. Jayaraman, A. B. Zaslavsky, P. Christen, and D. Georgakopoulos, "MOSDEN: An internet of things middleware for resource constrained mobile devices," in *Proc. Hawaii Int. Conf. Syst. Sci.*, 2013, pp. 1053–1062.
[24] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure—Physical sensor management with virtualized sensors on cloud computing," in *Proc. Int. Conf. Netw.-Based Inf. Syst.*, Sep. 2010, pp. 1–8.
[25] G. Fortino, M. Pathan, and G. Di Fatta, "BodyCloud: Integration of cloud computing and body sensor networks," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2012, pp. 851–856.
[26] N. Zingirian and C. Valenti, "Sensor clouds for intelligent truck monitoring," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2012, pp. 999–1004.
[27] G. Lewis, S. Echeverria, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *Proc. IEEE Military Commun. Conf.*, Oct. 2014, pp. 1440–1446.
[28] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st edition MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.
[29] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proc. 2nd ACM SIGCOMM Workshop Mobile Cloud Comput.*, 2013, pp. 15–20.
[30] X. Hu, L. Wang, Z. Sheng, P. TalebiFard, L. Zhou, J. Liu, and V. C. Leung, "Towards a service centric contextualized vehicular cloud," in *Proc. 4th ACM Int. Symp. Develop. Anal. Intell. Veh. Netw. Appl.*, 2014, pp. 73–80.
[31] T. Ali, M. Gheith, and E. Nasr, "Socially intelligent computing—A survey of an emerging field for empowering crowd," in *Proc. 9th Int. Conf. Informat. Syst.*, Dec. 2014, pp. PDC-102–PDC-108.
[32] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[33] C. Mei, D. Taylor, C. Wang, A. Chandra, and J. Weissman, "Sharing-aware cloud-based mobile outsourcing," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jun. 2012, pp. 408–415.
[34] B. Heintz, A. Chandra, R. Sitaraman, and J. Weissman, "End-to-end optimization for geo-distributed MapReduce," *IEEE Trans. Cloud Comput.*, vol. PP, no. 99, p. 1, Sep. 2014.
[35] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 4–4.
[36] W. Yuan and K. Nahrstedt, "Energy-efficient CPU scheduling for multimedia applications," *ACM Trans. Comput. Syst.*, vol. 24, no. 3, pp. 292–331, 2006.
[37] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 2001, pp. 50–61.
[38] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," in *Proc. 19th ACM Symp. Oper. Syst. Principles*, 2003, pp. 149–163.
[39] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York, NY, USA: Wiley, 1991.
[40] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.
[41] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, "Demystifying 802.11n power consumption," in *Proc. Int. Conf. Power Aware Comput. Syst.*, 2010, p. 1.
[42] J.-P. Vasseur and A. Dunkels, Interconnecting Smart Objects with IP: The Next Internet *Interconnecting Smart Objects with IP: The Next Internet*. San Mateo, CA, USA: Morgan Kaufmann, 2010.
[43] M. A. Razzaque and S. Dobson, "Energy-efficient sensing in wireless sensor networks using compressed sensing," *Sensors*, vol. 14, no. 2, pp. 2822–2859, 2014.
[44] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. Mccann, and K. Leung, "A survey on the IETF protocol suite for the Internet of Things: standards, challenges, and opportunities," *IEEE Wireless Commun. Mag.*, vol. 20, no. 6, pp. 91–98, Dec. 2013.
[45] J. Lee and N. Jindal, "Delay constrained scheduling over fading channels: Optimal policies for monomial energy-cost functions," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2009, pp. 1–5.
[46] M. Zafer and E. Modiano, "Delay-constrained energy efficient data transmission over a wireless fading channel," in *Proc. Inf. Theory Appl. Workshop*, Jan. 2007, pp. 289–298.
[47] J. Lee and N. Jindal, "Asymptotically optimal policies for hard-deadline scheduling over fading channels," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2482–2500, Apr. 2013.
[48] A. Mahanti, N. Carlsson, A. Mahanti, M. Arlitt, and C. Williamson, "A tale of the tails: Power-laws in internet measurements," *IEEE Netw.*, vol. 27, no. 1, pp. 59–64, Jan. 2013.
[49] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

Zhengguo Sheng received the BSc degree from the University of Electronic Science and Technology of China in 2006, and the MS (with distinction) and PhD degrees from Imperial College London in 2007 and 2011, respectively. He then joined Orange Labs. He is currently a lecturer at the School of Engineering and Informatics, The University of Sussex, United Kingdom. He is also the visiting faculty of University of British Columbia (UBC). Previously, he was with UBC as a research associate, and with France Telecom Orange Labs as the senior researcher and project manager in M2M/IoT. He also worked as a research intern with IBM T. J. Watson Research Center, and U.S. Army Research Labs. His current research interests cover IoT/M2M, cloud/edge computing, vehicular communications, and power line communication. He has published more than 40 International conference and journal papers. He is also the recipients of Auto21 TestDRIVE Competition Award 2014 and Orange Outstanding Researcher Award 2012. He is a member of the IEEE.

Chinmaya Mahapatra received the BTech degree in electronics and communication engineering from the National Institute of Technology, Rourkela, India, in 2009, and the MASc degree in electrical and computer engineering from The University of British Columbia (UBC), in 2013, where he is currently working toward the PhD degree with the Department of Electrical and Computer Engineering. Prior to joining UBC, he was a scientist with the Indian Defense Research Laboratory, and a systems engineer with the Ciena Research and Development Center, Ottawa, Canada. His current interests include the Internet of Things, embedded systems, sensor cloud, smartphone energy optimization, E-health, LTE, and Green communications. He is a student member of the IEEE.

Victor C. M. Leung is currently a professor of electrical and computer engineering and holder of the TELUS Mobility Research Chair at the University of British Columbia (UBC). His research interests include the areas of wireless networks and mobile systems, where he has coauthored more than 700 technical papers in archival journals and refereed conference proceedings, several of which had won best-paper awards. He is a fellow of the IEEE, the Royal Society of Canada, the Canadian Academy of Engineering, and the Engineering Institute of Canada. He is serving/has served on the editorial boards of the *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Computers*, *IEEE Transactions on Wireless Communications*, and *IEEE Transactions on Vehicular Technology*, *IEEE Wireless Communications Letters*, and several other journals. He has provided leadership to the technical committees and organizing committees of numerous international conferences. He was the recipient of an APEBC Gold Medal, NSERC Postgraduate Scholarships, a 2012 UBC Killam Research Prize, and an IEEE Vancouver Section Centennial Award.



Min Chen is currently a professor at the School of Computer Science and Technology, Huazhong University of Science and Technology. He is Chair of IEEE Computer Society Special Technical Communities on Big Data. He was an assistant professor at the School of Computer Science and Engineering, Seoul National University (SNU), from September 2009 to February 2012. He was the R&D director at Confederal Network Inc. from 2008 to 2009. He worked as a postdoctoral fellow at the Department of Electrical and Computer Engineering, University of British Columbia (UBC), for three years. Before joining UBC, he was a postdoctoral fellow at SNU for one and half years. His research interests include Internet of Things, machine to machine communications, body area networks, body sensor networks, e-healthcare, mobile cloud computing, cloud-assisted mobile computing, ubiquitous network and services, mobile agent, and multimedia transmission over wireless network, etc. He received best paper award from IEEE ICC 2012, and best paper runner-up award from QShine 2008. He has more than 180 paper publications. He has been a senior member of the IEEE since 2009.

Pratap Kumar Sahu received the PhD degree from National Central University, Taiwan. He is currently a postdoctoral researcher at the Network Research Lab in the Department of Computer Science and Operation Research, University of Montreal. His research interests include network coding, sensor networks, and vehicular ad hoc networks. Mostly, he works on various aspects of Intelligent Transportation Systems. He has many publications in various journals such as *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Sensors*, *Springer Telecommunication Systems* and conferences such as IEEE Globecom, IEEE ICC, and IEEE ICCCN. He is a member of the IEEE.