Indoor Temperature Monitoring Using Wireless Sensor Networks: A SMAC Application in Smart Cities

Shengbo Chen[a,b], Lanxue Zhang[a], Yuanmin Tang[c], Cong Shen[d], Roshan Kumar[e], Keping Yu[f,g,*], Usman Tariq[h], Ali Kashif Bashir[i]

[a]*School of Computer and Information Engineering, Henan University, Kaifeng 475001, China*
[b]*Shengmeng Technology Ltd, Jiangxi, 343000, China*
[c]*School of International Education College, Henan University, Kaifeng 475001, China*
[d]*Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904, USA*
[e]*School of Miami College, Henan University, Kaifeng 475001, China*
[f]*Global Information and Telecommunication Institute, Waseda University, Tokyo 169-8050, Japan*
[g]*Shenzhen Boyi Technology Company Ltd., Shenzhen 518125, China*
[h]*College of Computer Science and Engineering Prince Sattam bin Abdulaziz University, Saudi Arabia*
[i]*Department of Computing and Mathematics, Manchester Metropolitan University, United Kingdom*

*Corresponding author: Keping Yu (keping.yu@aoni.waseda.jp).

Indoor Temperature Monitoring Using Wireless Sensor Networks: A SMAC Application in Smart Cities

Abstract

SMAC (Social, Mobile, Analytics and Cloud) technologies aim to bridge the cyber, physical and social spaces. The use of wireless sensor networks to monitor indoor temperature is a typical application in smart cities. Rather than splitting the measured temperature and the design of a sensor network, a cyber-physical design approach is proposed by this paper for indoor temperature monitoring using wireless sensor network. The source sensors adopt sleep/wake scheduling, that is, source nodes wake up and sense the temperature periodically. The temperature data is sent to the cloud server via multi-hop relaying sensor nodes in an anycast way. Each sensor decides how to route packets based on its local information and dynamically adjust the sleep/wake duty cycle according to the sensed temperature: if the measured temperature is within normal range, the sensor wakes up infrequently to achieve higher energy efficiency; and vice versa. We first propose an optimal delay algorithm for anycast protocol. The simulation results show that our approach outperforms other heuristic schemes. Furthermore, we implement the proposed algorithm using TelosB sensors with TinyOS. Experiments demonstrate that the designed system can report a temperature anomaly within a small delay and achieve good long-term energy efficiency at the same time.

Keywords: SMAC, Intelligent Computing, Wireless Sensor Network, Temperature Monitoring

1. Introduction

SMAC (Social Mobile Analytics and Cloud) is a combination of disruptive technologies. These technologies are believed to drive next-generation data analysis, and infiltrate intelligence and decision-making into the physical world, thereby continuously shaping future cities and enhancing the human experience in real time. Social networks are related to data generated from social media platforms. The data shows the customer schedule on the subject of trends and the interests of customers [1, 2]. Mobile devices have become the largest business-building community because they allow users to update their profile, learn about the latest promotions, and track location by simply connecting to wireless signals [3]. Analytics is now a cornerstone of innovation because it can make intelligent predictions about customer behavior in a relatively short period of time by looking at billions of customer data records available in databases [4]. Cloud computing makes it easy for companies to access data by renting data from cloud providers without investing millions of dollars in building a data warehouse [5].

Some works have applied SMAC technologies to smart home, smart agriculture and other fields, which achieved impressive results. The authors [6] use Internet of Things(IoT) and sensors to implement a long-term environmental monitoring system. The detailed design and components of the platform take into account the characteristics of low application cost, large number of sensors, fast deployment, long life, low maintenance, and good service quality. Malche [7] describes Frugal Labs IoT Platform (FLIP) for building IoT enabled Smart Home and introduces FLIP architecture with implementation of Smart Home services. The system continuously monitors home air quality and sends alerts to users if there are health risks. It also improves security because users can monitor all activities in their home. In addition, the system makes better use of energy and resources through smart lighting and smart air conditioning systems. In [8], the authors propose an environmentally sustainable home automation system model and control home IoT appliances through an internet-connected device. This model utilizes Android as the basic platform to ensure that the system can be widely used by customers, while the cost of the system is quite low. A new wireless mobile robot is designed and implemented based on the IoT in [9]. The main feature of this intelligent wireless robot is that it can perform tasks such as humidity sensing, startle birds, spray pesticides, move forward/backward and turn on/off electric motors. The robot has been shown to be very useful for intelligent agricultural systems. Chatap [10] investigates the latest cluster protocols for heterogeneous wireless sensor networks (WSN), and concludes through experiments that the proposed protocol has better performance than other existing cluster protocols in a heterogeneous WSN environment.

While tons of data is collected, it requires high computation capacity. As cloud storage servers store a large amount of data [11], cloud computing plays a key role in processing these high-volume data. On the other hand, with the increasing computing capacity and communication capabilities, fog-edge computing, as an important and effective supplement of cloud computing, has been widely used to deal with the local real-time data. In fog-edge computing, cloud elastic resources are extended to the edge of the network, such as wireless sensors, mobile devices, smart objects and other IoT devices [12, 13, 14, 15, 16], in order to reduce the latency and network congestion.

As an important component for SMAC in smart cities, wireless sensor networks have been widely adopted for monitoring the physical environment. Temperature monitoring is one of the most successful applications of wireless sensor network. It becomes even more important with the fast growth of cloud computing, because cloud servers have to run in appropriate temperature range for purpose of achieving near-optimal capability and security. Because of the low maintenance and installation cost, wireless sensor networks have received lots of attention for monitoring indoor temperature [17]. Especially, densely deployed wireless sensors can be used to scale the temperature in a building without requiring a fixed wired connection [18]. The sensor networks can sense the temperature, which can be transmitted to a controller. If a temperature anomaly occurs, the cloud server will trigger an alarm.

In the existing sensor networks, on behalf of assuring the temperature reported anomaly in time, the sensors oblige to measure the environment often, which causes high energy consumption. Additionally, if the sensor networks struggle to have a better energy efficiency by prolonging the sleep period, it may not be capable to perceive the temperature anomaly in time, which can be unaccepted in certain applications. Therefore, it is distinct that there is an elementary tradeoff between detection delay and energy efficiency—the greater the duty cycle, the lower the energy consumption, but the longer the detection latency. The vital problem is how sensors regulate the duty cycles and determine routing based on local data and information.

In this paper, we propose an indoor temperature monitoring system using a cyber-physical method, where both the restrictions of wireless sensor networks (i.e., energy consumption) and the demands of the indoor temperature monitoring (i.e., low detection latency) are collectively considered. Exploring the *graduality* characteristic of the variations in temperature is the key idea. Particularly, the sensors are designed to regulate their own duty cycles according to the measured temperatures. The sensors will wake up rarely to achieve a high energy efficiency when the measured temperature is normal. On the other hand, if the temperature is anomalous, the sensor nodes reduce their duty-cycles in order that the temperature anomaly alarm will be triggered in time.

Sleep/wake scheduling in wireless sensor networks has been widely investigated in the literature; look the recent papers [19, 20, 21, 22, 23, 24] and the references therein. [25, 26, 27] have proposed synchronized sleep/wake protocols. But, such synchronization procedures cause attached communication overhead and result in low energy efficiency. [28, 29, 30] propose asynchronous sleep/wake scheduling protocols. In these protocols, every node wakes up independently for purpose of saving energy. However, additional delay produces at every hop because each node obliges to wait for its next-hop node waking up in order that the packet can be transmitted. In this paper, to solve the high detection latency problem, we employ an asynchronized sleep/wake protocol, and adopt anycast packet-forwarding schemes (a.k.a. opportunistic forwarding schemes). Such approach shows some characteristics resembling to both unicast and multicast, and has been shown to reduce the end-to-end latency efficiently. In particularly, nodes sustain multiple candidates of next-hop nodes and transmit their packets to the candidate node that wakes up first in anycast packet-forwarding schemes. Therefore, the one-hop delay can be reduced substantially by using anycast packet-forwarding scheme. Differing from the NQA algorithm proposed in [31] which can make full use of the preprocessing time to improve the efficiency performance, our designed indoor temperature monitoring system implements anycast, which not only has low detection latency of the temperature anomaly, but also achieves better energy efficiency.

The main contributions of this paper can be summarized as follows.

- We put forward a cyber-physical design of indoor temperature monitoring system with wireless sensor networks, in which the configuration of sensor nodes is dynamically regulated according to the measured temperature. This design employs an asynchronized sleep/wake protocol, and adopts anycast packet-forwarding schemes. We first propose an delay optimal scheme subject to network lifetime constraint, which is proven by both analysis and simulations.

- We illustrate an implementation of this architecture using TelosB sensors and the TinyOS operation system, and a theoretical analysis of the detection delay is also provided.

- We implement the proposed solution in a real-world platform, and experiment results are reported. In particular, latency and energy consumption measurements from these experiments prove the advantage of our approach.

2. System Model

We consider a wireless sensor network with N nodes. Assume there are multiple source nodes $N_{s_i}, N_{s_i} \in \mathbb{S}$, and multiple destination nodes $N_{d_i}, N_{d_i} \in \mathbb{D}$, where $\mathbb{S}$ and $\mathbb{D}$ are the set of source nodes and destination nodes. Each source node can detect events by sensing the environment, such as monitoring temperature. When an event is detected, the source node will generate a packet and transmit to the destination node via multi-hop replay. We assume that all destination nodes are homogeneous, that is, a packet can be transmitted to any destination node rather than a particular sink. Note that our solution can be easily extended to the case when destination nodes are heterogeneous.

2.1. Sleep-wake Scheduling

All sensor nodes use sleep-wake scheduling to improve the energy efficiency. When a node has a packet to transmit, it keeps sending beacon signals until it receives an ACK from some relay node. When a node wakes up from the sleep mode, it performs sensing immediately. If it senses a beacon from upstream node, it replies an ACK. Then the packet will be transmitted. If it does not detect any beacon, the node goes back to sleep immediately. The sleep period X_i for each node i is assumed to be an independent exponentially distributed random period of time with mean λ_i. In this paper, we assume the event is rare and most energy is consumed for wake-up. Therefore, when node i wakes up more frequently, that is, λ_i is larger, the node consumes energy faster, while achieving a smaller delay. On the other hand, when λ_i is smaller, the node will have a larger delay while guaranteeing a better energy efficiency. Hence, we define the expected lifetime for node i as

$$L_i = \frac{E_i}{\lambda_i e_i}, \ \forall i \in (1, \cdots, N), \tag{1}$$

where E_i is the total energy available for node i, and e_i is the expected energy consumption rate for each wake-up period. We assume both E_i and e_i are known in advance.

2.2. Anycast Protocol

Each node i has multiple candidate relay nodes, which we call forwarding set $\mathbb{F}_i$. The node picks the first wake-up node in $\mathbb{F}_i$ to transmit the packet. Since we assume the sleep periods to be exponentially

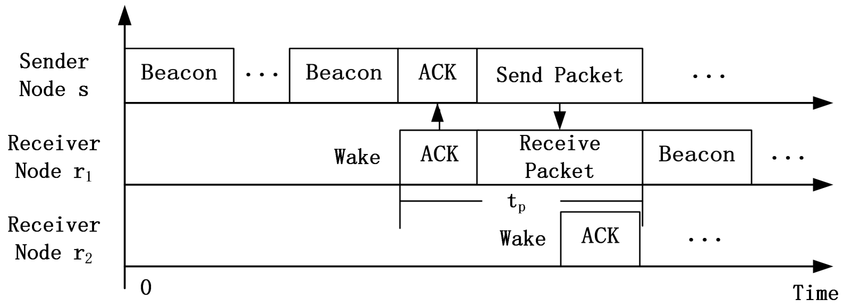

Figure 1: Anycast model

distributed for all nodes, the time before the first node wakes up is also exponentially distributed with parameter $\sum_{j\in\mathbb{F}_i}\lambda_j$. This means that the delay for next hop will be greatly reduced. Fig. 1 shows an example for anycast protocol, where one sender node has two nodes in its forwarding set. Since node r_1 wakes up before node r_2, the sender node sends the packet to node r_1 after receiving ACK.

We define the expected delay from node i to destination nodes as D_i. Denote p_{ij} as the probability that node j wakes up first in the forwarding set and T_p is the packet transmission time. The delay D_i consists of three components: the time period for the first forwarding node wakes up, the packet transmission time to the forwarding node, and the expected delay for the forwarding node. Hence, we will have

$$D_i = \frac{1}{\sum_{j\in\mathbb{F}_i}\lambda_j} + T_p + \sum_{j\in\mathbb{F}_i} p_{ij} D_j. \tag{2}$$

Also because the sleep period is exponentially distributed, we have

$$p_{ij} = \frac{\lambda_j}{\sum_{k\in\mathbb{F}_i}\lambda_k}. \tag{3}$$

2.3. Problem Formulation

The objective is to minimized the expected delay from each source node to the destinations, while guaranteeing the lifetime of each node i is above some threshold ϵ_i. Hence, the problem can be formulated as

$$\text{Problem } \mathbf{A:} \quad \min_{\vec{\lambda}_i, \vec{\mathbb{F}}_i} \quad D_i$$
$$\text{subject to} \quad L_i \geq \epsilon_i, \; \forall i. \tag{4}$$

3. Optimal Anycast Algorithm

In this section, we will first propose the optimal anycast algorithm, that is, the solution to Problem A. Then we verify the optimality through simulations.

3.1. Algorithm Description

We first describe our algorithm as follows:

OAA: Optimal Anycast Algorithm

(1) Each node i initializes its delay by $D_i = \infty$, $\lambda_i = \frac{E_i}{\epsilon_i e_i}$, and forwarding set $\mathbb{F}_i = \mathcal{C}_i$, where $\mathcal{C}_i$ is the set of all nodes that are in the communication range of node i.

(2) For each iteration, each node i updates its delay by Eqn. (2) and Eqn. (3).

(3) For each iteration, each node i updates its forwarding set by

$$\mathbb{F}_i = \{j \in \mathcal{C}_i | D_j \leq D_i - T_p\}. \tag{5}$$

(4) If both D_i and $\mathbb{F}_i$ keep the same from iteration $h-1$ to h for each node i, the algorithm stops. Otherwise, return to Step 2.

The optimality of ***OAA*** is characterized by the following theorem:

Theorem 1*:* The delay for each node is minimized under ***OAA***.

Proof. We need to prove that the optimal forwarding set should be selected according to Eqn. (5).

Assume that the optimal forwarding set $\hat{\mathbb{F}}_i$ is not selected according to Eqn. (5). There will be two cases.

Case 1: $\hat{\mathbb{F}}_i \supset \mathbb{F}_i$. This means that some node $j \in \hat{\mathbb{F}}_i$ has its delay $D_j > D_i - T_p$. Clearly, according to Eqn. (2), the delay D_i can be reduced if this node j is removed from the forwarding set.

Case 2: $\hat{\mathbb{F}}_i \subset \mathbb{F}_i$. This means that some node $k \notin \hat{\mathbb{F}}_i$ has its delay $D_k < D_i - T_p$. We will show that if we include node k in the forwarding set, the new delay $\hat{D}_i$ is smaller than D_i. According to Eqn. (2) and (3), we have

$$\begin{aligned}\hat{D}_i - D_i &= \frac{\sum_{j \in \mathbb{F}_i \cup k} \lambda_j D_j + 1}{\sum_{j \in \mathbb{F}_i \cup k} \lambda_j} - \frac{\sum_{j \in \mathbb{F}_i} \lambda_j D_j + 1}{\sum_{j \in \mathbb{F}_i} \lambda_j} \\ &= \lambda_k \frac{\sum_{j \in \mathbb{F}_i} \lambda_j D_k - \sum_{j \in \mathbb{F}_i} \lambda_j D_j - 1}{\sum_{j \in \mathbb{F}_i \cup k} \lambda_j \cdot \sum_{j \in \mathbb{F}_i} \lambda_j}\end{aligned} \tag{6}$$

From Eqn. 2, we have

$$\sum_{j \in \mathbb{F}_i} \lambda_j D_j = (D_i - T_p) \cdot \sum_{j \in \mathbb{F}_i} \lambda_j - 1. \tag{7}$$

Combining Eqn. 6 and 7, we have $\hat{D}_i - D_i < 0$ given that $D_k < D_i - T_p$. This means that the delay can be reduced if we include node k in the forwarding set.

Hence, we conclude that the optimal forwarding set should be selected according to Eqn. (5). □

3.2. Simulations

3.2.1. One source to one destination

We randomly generate a network with 100 nodes containing one source node and one destination node. As shown in Fig. 2, the source node and the destination node are at the bottom-left and top-right corner, respectively. We assume that all nodes have the same lifetime constraint $\epsilon_i = 1, \forall i$, i.e., $L_i \geq 1$, but with different energy availability $\frac{E_i}{e_i}$, which is in the range [5,15]. Thus, according to Eqn. (1), the sleep period for node i is generated following exponentially distribution with parameter $\lambda_i = \frac{E_i}{e_i}$. The transmission delay T_p is assumed to be 0.1 second.

We first use ***OAA*** to calculate the delay and the forwarding set for each node. Then for each run of the simulation, one packet is generated and transmitted to the destination via anycast. The red curve in Fig. 2 demonstrates an example of the routing path, while the green dash lines are the forwarding

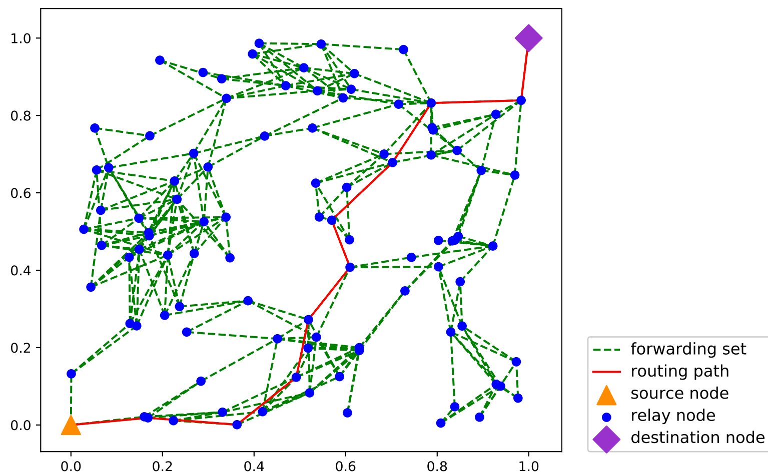

Figure 2: Example for the anycast routing with one source and one destination

sets. We repeat running the simulation for 10000 times and average the delays for the source node. We obtain the average delay of source node, which is 1.754 second, while the calculated expected delay is 1.753 second. We can see that the simulation result and the analytical result are almost identical.

Besides the anycast routing, we also simulate the deterministic routing scheme, where each node chooses the node with the minimum delay in its forwarding set as the next hop. In this case, the average source-to-end delay is 9.0210 second, which is 80.6% higher than the delay for anycast.

We also compare the delays under different schemes, as shown in Fig. 3. The x-axis is the lifetime constraint ϵ_i, and the y-axis is the simualted delay. The blue curve is the delay of our $\boldsymbol{OAA}$ algorithm. The orange curve, labeled as "Deterministic routing" represents the scheme which always selects the node with the smallest delay in the forwarding set. Similarly, the green and red curves, represent the schemes that always choose 2 and 3 nodes in the forwarding set, respectively. It can be seen that the delay of our $\boldsymbol{OAA}$ algorithm is smaller than all other heuristic schemes.

3.2.2. One source to multiple destinations

The settings of this simulation are the same as the previous simulation except that there are three destination nodes, as shown in Fig. 4. The average delay for source node over 10000 runs is 1.107 second, while the analytical expected delay is 1.110 second for anycast routing. Similarly, the average delay for deterministic routing is 1.350 second, which is 17.8% higher than anycast routing.

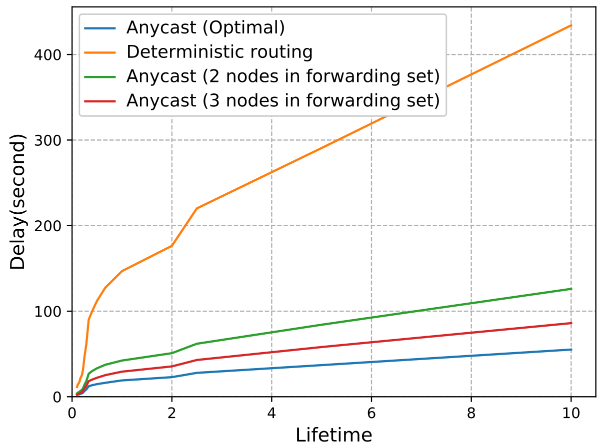

Figure 3: The delay versus lifetime under different schemes

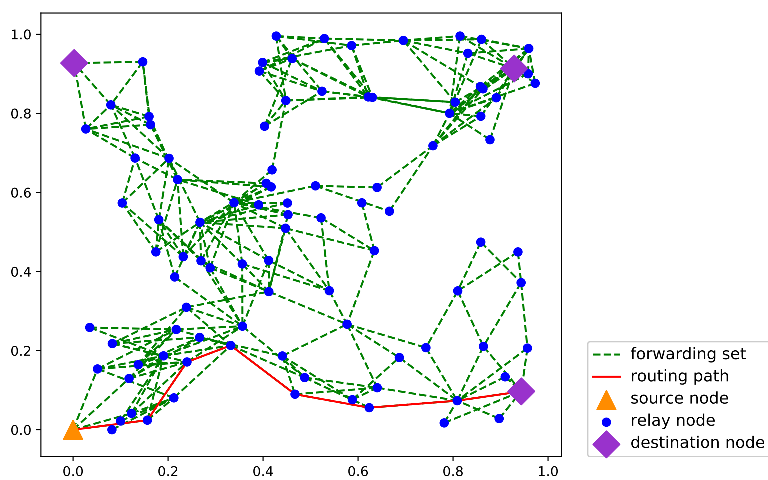

Figure 4: Example for the anycast routing with one source and three destinations

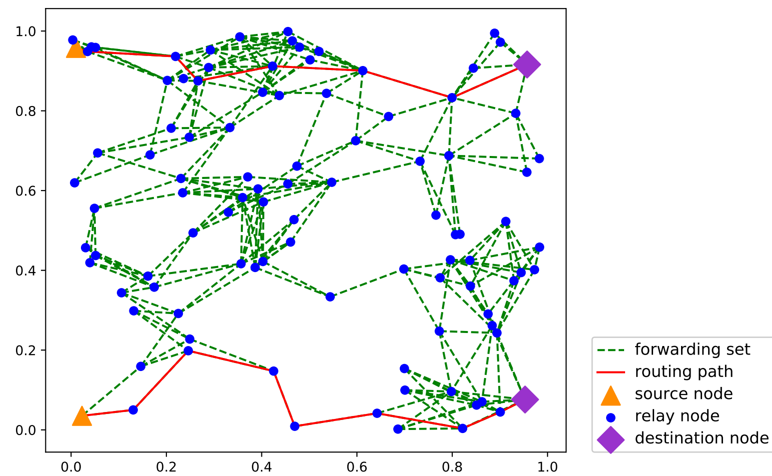

Figure 5: Example for the anycast routing with two sources and two destinations

3.2.3. Multiple Sources to Multiple Destinations

In this part, we use two source nodes and two destination nodes, as shown in Fig. 5, while other settings remain the same. The average delays for both source nodes over 10000 runs are 1.348 second and 1.203 second, respectively, while the analytical expected delays are 1.420 second and 1.204 second. Similarly, the average delays for deterministic routing are 1.682 second and 1.511 second, respectively, which shows that anycast routing achieves better average delays by 19.9% and 20.4% than deterministic routing, respectively.

4. Experiment Design

4.1. Hardware Platform

Our system is built based on the TelosB sensors with TinyOS operation systems [32]. TelosB has a MSP430 microcontroller, as well as a CC2420 communication chipset using IEEE 802.15.4. It has many superiorities, such as high data rate and low power rate. And the USB of TelosB can be used for programming, debugging and collecting data. We show the TelosB hardware platform that we use in this work in Fig. 6. In particular, SHT11 is the component for temperature sensing. SHT11 also combines its sensor with signal-processing element on a footprint and a fully calibrated digital output is provided. The CMOSens@ technology provides good stability and reliability, which is connected to TelosB through I^2C.

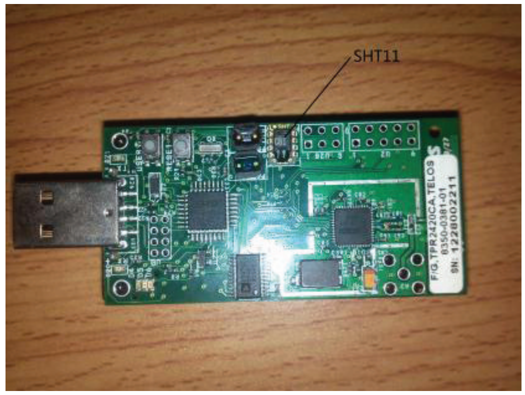

Figure 6: TelosB with SHT11.

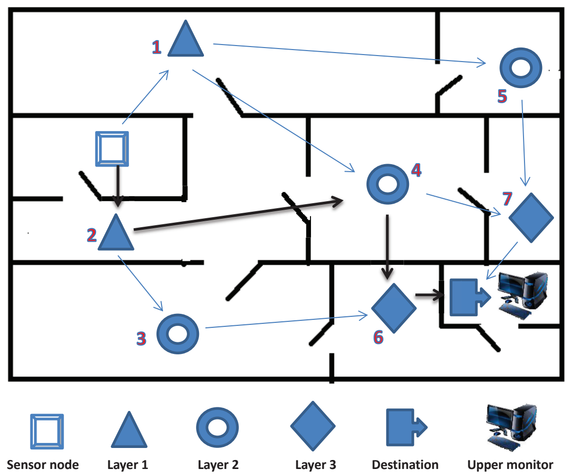

Figure 7: Illustration of the network topology.

TinyOS is an open-source operation system designed for wireless sensor network, which can support complex, concurrent programs requiring low memory and low-power operations [33]. It has a lightweight thread technology, component-based architecture, active message communication model, and event-driven execution model. Component-based architecture enables quick development, as its library includes lots of key elements, like wireless protocols, sensor drivers, distributed services, and data collecting tools. The event-driven execution model can deal with greatly concurrent events and realize good energy efficiency.

4.2. Network Topology

In our implementation, we deploy nine sensors within six rooms, where the topology is shown in Fig. 7. The network consists of one source node, one destination node and seven relay nodes. The source node uses sleep-wake scheduling and measures the temperature. The destination node requires to receive the measured temperature from the source and then transmit to the server. As the destination node is out of the transmitting range of the source node, relaying nodes are needed for packets forwarding. The relay nodes are categorized into distinct layers, according to the hops away from the source. For the network in Fig. 7, node 1 and 2 can directly hear the packets from the source, thus they are called layer 1 nodes. Analogously, layer 2 consists of nodes 3, 4 and 5, and layer 3 includes nodes 6 and 7. Notice that the forwarding set is determined by the number of hops from the source, which is optimal in this setting.

When powering on the network, the initialization is done first in order that relay nodes are categorized to diverse layers. A control frame including an 8-bit content *layernum* is used during the initialization process. Particularly, once the source node wakes up, it will broadcast this packet including *layernum*=0x00 periodically for a long time, e.g., 5 second, to ensure that all adjacent nodes received it. The source node then broadcasts *layernum*=0xff to claim the accomplishment of initialization. After the control packet is received by a node, it will analyze the payload and make decision consequently. More explicitly, if received *layernum* is not the same as 0xff and smaller than a local variable *mylayer* (the default value is 0xfc), the node sets *mylayer* to be *layernum*+1. After layernum=0xff is received, it begins to broadcast its own control packet with *layernum*=*mylayer* in a similar way as the source. This procedure keeps iterating until each node gets its own layer. The procedure is shown in Fig. 8. Node 1 and 2 figure out that they are layer 1 after receiving the control frame from the source. Equally, Node 3, 4 and 5 know that they are layer 2, and Node 6 and 7 are layer 3.

In the data transmission, the transmitter's layer information is included in the beacon. Once a beacon is detected, the relay node extracts the layer number and drops the packets with a larger layer number. When the control packet with a layer number is received by the destination node, it will broadcast 0xfd to claim that it is the destination. Table 1 summarizes the functionalities of *layernum*.

Table 1: Functionality of *layernum* in control packet.

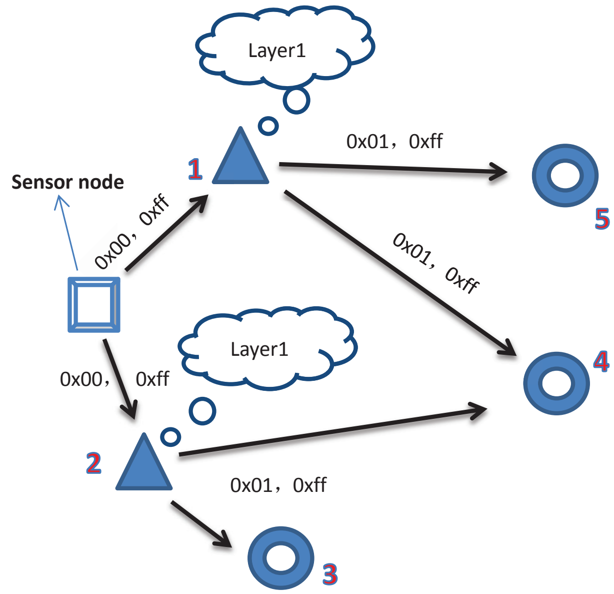

Figure 8: Illustration of the layer division process.

layernum	functionality
0x00-0xfb	$mylayer = layernum + 1$ if $mylayer < layernum$
0xfd	it is the destination node
0xfe	one of the following layer is down, broadcast again to check
0xff	start broadcasting

5. Cyber-Physical System Design

5.1. Sleep/Wake Scheduling

In order to reduce the energy consumed in wireless sensor networks, all nodes adopt the Low Power Listening (LPL) [34], which is a MAC-layer technique. Nodes wake up periodically to sense the channel. LPL has been realized in TinyOS.

We adopt a dynamic *Temperature-Determined Duty Cycle* (TDDC) scheme for all nodes in the system. Particularly, after the source sensor measures the temperature or the relay sensors parsed the temperature, they regulate the sleep-wake cycle according to a pre-determined function between the

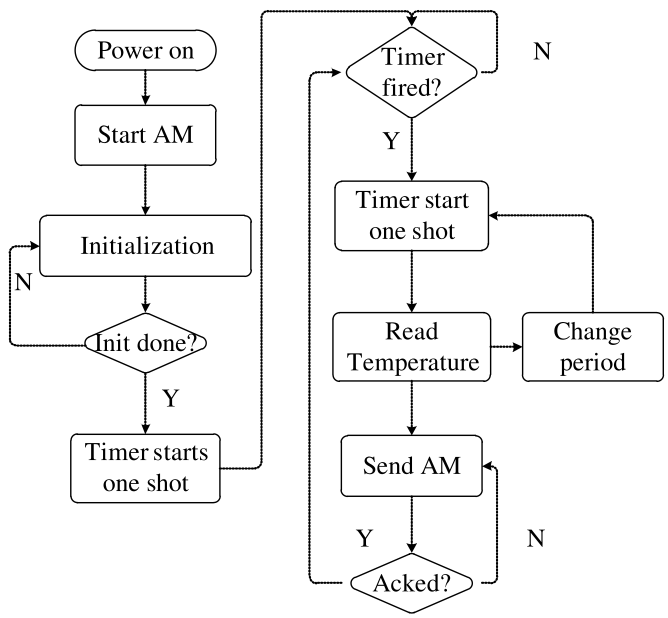

Figure 9: Flow chart of the sensor node protocol.

duty cycle parameter and the temperature. If the temperature is abnormal, the nodes can reduce the sleep period in order to wake up more frequently for monitoring the temperature.

5.2. Anycast Protocol

In this subsection, we list the protocol by different sensor node categories.

5.2.1. Source Node

Fig. 9 illustrates the protocol flow chart for a source node. Once the sensor wakes up, we initiate a timer, which aims to determine the period to measure temperature. When the timer ends, the sensor node measures the temperature, regulates its duty cycle length, and broadcasts the packet. After the temperature is measured, the sensor keeps transmitting the beacon until receiving an ACK from layer 1 node. It will then go back to sleep, the length of which is determined by its current measured temperature.

5.2.2. Relay Sensor

The protocol flow chart for a relay node is given in Fig. 10. A relay sensor wakes up based on its own duty-cycle period. When the relay node wakes up, it senses the channel immediately. If there are no beacon, the relay node goes back to sleep. Otherwise, relay node will send an ACK and receive the

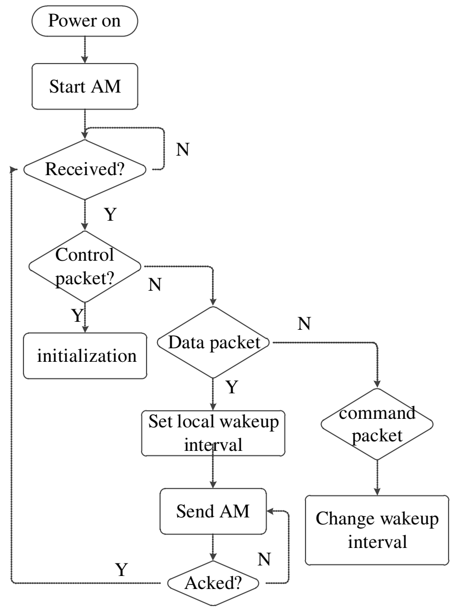

Figure 10: Flow chart for the relay node protocol.

packet.

There are three packet categories. *Data packets* are the ones from the source, in which includes a temperature value (16 bits), a counter number (8 bits), a layer information (8 bits), and a sample period (16 bits). The sample period is to determine the duty cycle, i.e., the interval for wake up. Transmitting a packet backwards is prevented by layer information. The counter aims to perceive possible duplication of the packets. More specifically, the new packet will be dropped if the counter is same as the counter for some previously received packet. *Control packets* aim for the initialization process, where a layer number (8 bits) is embedded. *Command packets* are from the destination sensor. The target is to adjust the wake-up interval of relay nodes. A counter number (8 bits) and a parameter (8 bits) are included in the message.

5.2.3. Destination Sensor

The destination node and the server (a PC) are connected using USB. Since it is powered by USB, it keeps monitoring the channel. All received packets are forwarded to the server for post-processing.

5.3. Delay Analysis

Source-to-end delay is a key factor in designing alarming system. When temperature anomaly happens, it requires a small delay. In our system, the delay is primarily from relay nodes which adopts

LPL strategy.

In this subsection, we will obtain the source-to-end delay theoretically, which sheds light on the system performance. Assume that the sleeping periods for all the relay nodes are an independent and identically distributed (i.i.d) uniform distribution, then we have the probability density function for any node.

Assume that there are n nodes in the next layer, whose time to wake-up are donated as $X_1, X_2, \cdots, X_n$, respectively. Then, the delay for the first wake up node is given by $X = min(X_1, X_2, \cdots, X_n)$, and the distribution function is derived as:

$$\begin{aligned} F_X(x) &= P(X \leq t) = 1 - P(X > t) \\ &= 1 - P(X_1 > t, X_2 > t, \cdots X_n > t) \\ &= 1 - \Pi_{i=1}^{n} P(X_i > t) \\ &= 1 - \Pi_{i=1}^{n} [1 - F_{X_i}(x)] \end{aligned}$$

Thus, we have

$$F_X(x) = 1 - [1 - F_{X_i}(x)]^n = \begin{cases} 1, & x \geq t_m \\ 1 - (1 - \frac{x}{t_m})^n, & 0 < x < t_m \\ 0, & x \leq 0 \end{cases}$$

and we can derive the pdf of X as

$$f_X(x) = \begin{cases} \frac{n}{t_m}(1 - \frac{x}{t_m})^{n-1}, & 0 < x < t_m \\ 0, & \text{else} \end{cases}$$

Therefore,we can write the expectation of this layer delay as

$$
\begin{aligned}
E_n(X) &= \int_{-\infty}^{+\infty} x f_X(x) dx \\
&= \int_0^{t_m} x \frac{n}{t_m}(1 - \frac{x}{t_m})^{n-1} dx \\
&= \frac{t_m}{n+1}
\end{aligned}
$$

Consider the network in Fig. 7, since there are two layer 1 nodes connecting to the source sensor, the delay is $E_2(X) = \frac{t_m}{3}$. Similarly, the delay to layer 2 is $\frac{1}{2} \times \frac{t_m}{3} + \frac{1}{2} \times \frac{t_m}{3} = \frac{t_m}{3}$, and the delay to layer 3 is $\frac{1}{4}E_1(X) + \frac{1}{2}E_2(X) + \frac{1}{4}E_1(X) = \frac{1}{2} \times \frac{t_m}{3} + \frac{1}{2} \times \frac{t_m}{2}$. Thus, the total delay is

$$E(X) = \frac{5}{2} \times \frac{t_m}{3} + \frac{1}{2} \times \frac{t_m}{2} = \frac{13}{12} t_m.$$

It is noticed that t_m is a function of measure temperature in our design. When temperature raises, t_m will decline, which result in a smaller source-to-end delay.

6. Implementation

6.1. Temperature Record

To mimic a anomalous temperature, a lighter is used to increase the temperature. The flame first gets closer to the source node in order that the temperature gets higher, and then remove away from the source. The recorded temperature is illustrated in Fig. 11. Each dot in the figure represents a data packet, the blue line denotes the time stamps when the controller receive a packet. We can see that the data packets are generated more frequently during high temperature, which can be expected from our design. An alarm will be triggered and displayed if the temperature is higher than an threshold (set to be 50 degree celsius in our design).

6.2. Relationship between Delay and Temperature

The source-to-end delay is measured and compared to the theoretical analysis in previous section. Note that the wake up period of relay nodes t_m is a function of the temperature: t_m declines when the temperature increases, and vice versa. To efficiently and effectively measure the delay, the destination

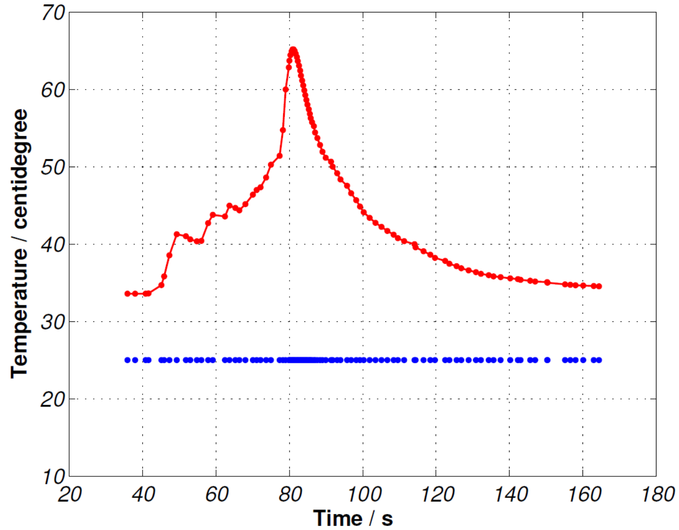

Figure 11: An example for temperature record.

node is allowed to transmit packets back to the source node. The packets from the source include a counter number (named *count*). When the source node forwards a packet, it saves the current time in *systime*[*count*]. When a packet is received by the destination, an ACK packet including the counter number will be forwarded back to the source node through a quick backward path, which consists of all relay nodes monitoring all the time rather than using LPL. So, the delay for the back path is relatively small and is considered negligible comparing to the forward-path delay. The sensor node reads the timer and calculates the delay when it receives the data packet.

```
btrpkt = (AckMsg*)payload;
count = btrpkt->ct;
time_now =  Timer1.getNow();
delay[count] = time_now-systime[count];
```

We report the average measured delay under different temperatures in Fig. 12, using the network topology and measurement strategy mentioned above. The theoretical result is also shown for comparison.

We can observe from Fig. 12 that the theoretical delay is always 100ms smaller than the measured delay, but the shape of both matches well. The key reason for the difference is as follows: it takes time for sending, receiving and processing packets, but they are not taken into account in the mathematical

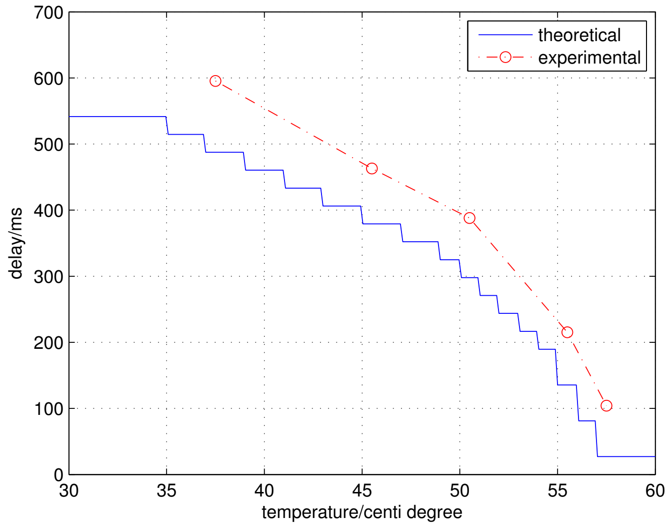

Figure 12: Delay versus temperature.

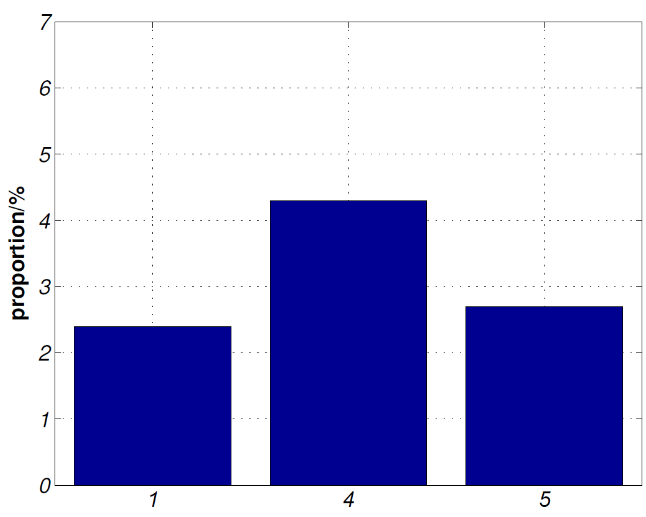

Figure 13: Proportion of the wake-up state of node 1, 4 and 5.

analysis. Moreover, the delay will get shorter when temperature increases, and the overall trend coincides with our mathematical analysis.

6.3. Wake-up State Proportion

As the relay sensors use the LPL strategy, they only wake up for a short time period to sense the channel. To transmit a packet, it needs a longer period since it has to wait for a wake-up node in its forwarding set. Since the time for sending a packet is much larger than the time for wake up, we neglect the latter one and only measure the former in this experiment.

When the sensor receives a packet, we record a system time stamp as *starttime* (32 bits). After the node receive an ACK, the current time will also be recorded as $finishtime$ (32 bits). Hence, we can derive the wake-up delay for each packet by $finishtime - starttime$. It is noted that the delay is less than the true value, since the RF cannot be stopped immediately when the transmission finishes. Nevertheless, since the unmeasured delay is much smaller than the wake-up delay, ignoring it will not result in any meaningful mismatch.

We test node 1, 4 and 5 for 30 minutes when the temperature is 55.5 degrees, and the result is plotted in Fig. 13. As we can see, the ratio of the wake-up state is around 2% - 5%. In particular, for each 100 seconds, node 1 wakes up for 2.4 seconds; node 4 wakes up for 4.3 seconds and node 5 for 2.7 seconds. We thus conclude that our design has a small proportion for wake-up and high the energy efficiency compared to the case without LPL.

7. Conclusion

In this work, we develop a cyber-physical-social design approach for temperature monitoring with wireless sensor networks. The source nodes adopt sleep/wake scheduling, that is, wake up and sense the temperature with duty cycles, and forward the data to the destination through multi-hop relaying nodes with anycast protocol. We first show an optimal delay algorithm, which is proven analytically and via simulations. We further implement this anycast scheme on a sensor platform. We dynamically adjust the period of sleep/wake duty-cycle based on the measured temperature. Particularly, when the measured temperature is in the normal range, the sensor nodes wake up infrequently in order to achieve high energy efficiency. On the other hand, if the sensed temperature increases closer to a threshold, the sensors wake up more frequently so that the an alarm can be triggered in time. We implement our design using TelosB sensors with TinyOS, and we show that experimental delay matches our mathematical analysis. Besides, it can detect and report a temperature alarm with a small delay while achieving high energy efficiency over a long duration.

ACKNOWLEDGMENT

This work is extension of our previous work published in the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, 2017 [35]. This work was supported by Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) under Grant JP18K18044, the Ji'An Finance & Science Foundation under Grant No. [2019]55, and the Grant for Ji'An Key Lab of computer-aided diagnosis for mental disease.

References

[1] H. Elazhary, Internet of things (iot), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions, Journal of Network and Computer Applications 128 (2019) 105 – 140.

[2] J. Loy-Benitez, Q. Li, K. Nam, C. Yoo, Sustainable subway indoor air quality monitoring and fault-tolerant ventilation control using a sparse autoencoder-driven sensor self-validation, Sustainable Cities and Society 52 (2020) 101847.

[3] T. Guelzim, M. Obaidat, B. Sadoun, Chapter 1 - introduction and overview of key enabling technologies for smart cities and homes, in: M. S. Obaidat, P. Nicopolitidis (Eds.), Smart Cities and Homes, Morgan Kaufmann, Boston, 2016, pp. 1 – 16.

[4] M. K. Saggi, S. Jain, A survey towards an integration of big data analytics to big insights for value-creation, Information Processing & Management 54 (5) (2018) 758 – 790, in (Big) Data we trust: Value creation in knowledge organizations.

[5] S. Jegadeesan, M. Azees, P. M. Kumar, G. Manogaran, N. Chilamkurti, R. Varatharajan, C.-H. Hsu, An efficient anonymous mutual authentication technique for providing secure communication in mobile cloud computing for smart city applications, Sustainable Cities and Society 49 (2019) 101522.

[6] D. Punniamoorthy, V. S. Kamadal, B. Srujana Yadav, V. S. Reddy, Wireless sensor networks for effective environmental tracking system using iot and sensors, in: 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on, 2018, pp. 66–69.

[7] T. Malche, P. Maheshwary, Internet of things (iot) for building smart home system, in: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 65–70.

[8] S. L. S. Sri Harsha, S. C. Reddy, S. P. Mary, Enhanced home automation system using internet of things, in: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 89–93.

[9] K. L. Krishna, O. Silver, W. F. Malende, K. Anuradha, Internet of things application for implementation of smart agriculture system, in: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 54–59.

[10] A. Chatap, S. Sirsikar, Review on various routing protocols for heterogeneous wireless sensor network, in: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 440–444.

[11] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, Y. Zhang, Blockchain empowered arbitrable data auditing scheme for network storage as a service, IEEE Transactions on Services Computing.

[12] F. Al-Turjman, A. Malekloo, Smart parking in iot-enabled cities: A survey, Sustainable Cities and Society 49 (2019) 101608.

[13] S. E. Bibri, The iot for smart sustainable cities of the future: An analytical framework for sensor-based big data applications for environmental sustainability, Sustainable Cities and Society 38 (2018) 230 – 253.

[14] Z. A. Khan, Using energy-efficient trust management to protect iot networks for smart cities, Sustainable Cities and Society 40 (2018) 1 – 15.

[15] Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, Y. Zhang, A blockchain-based nonrepudiation network computing service scheme for industrial iot, IEEE Transactions on Industrial Informatics 15 (6) (2019) 3632–3641.

[16] M. Diaz, C. Martin, B. Rubio, State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing, Journal of Network and Computer Applications 67 (2016) 99 – 117.

[17] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (4) (2002) 393 – 422.

[18] J. Li, J. He, A. Arora, Thermonet: fine-grain building comfort-efficiency assessment, in: Proceedings of the 3rd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, 2011, pp. 43–44.

[19] J. Kim, X. Lin, N. Shroff, P. Sinha, Minimizing delay and maximizing lifetime for wireless sensor networks with anycast, IEEE/ACM Transactions on Networking 18 (2) (2010) 515–528.

[20] Y. Z. Zhao, C. Miao, M. Ma, J. B. Zhang, C. Leung, A survey and projection on medium access control protocols for wireless sensor networks, ACM Computing Surveys 45 (1) (2012) 1–37.

[21] A. Bachir, M. Dohler, T. Watteyne, K. Leung, MAC essentials for wireless sensor networks, IEEE Communications Surveys and Tutorials 12 (2) (2010) 222–248.

[22] S. K. Mohapatra, P. K. Sahoo, S.-L. Wu, Big data analytic architecture for intruder detection in heterogeneous wireless sensor networks, Journal of Network and Computer Applications 66 (2016) 236 – 249.

[23] S. Tian, S. M. Shatz, Y. Yu, J. Li, Querying sensor networks using ad hoc mobile devices: A two-layer networking approach, Ad Hoc Networks 7 (5) (2009) 1014 – 1034.

[24] M. Abujubbeh, F. Al-Turjman, M. Fahrioglu, Software-defined wireless sensor networks in smart grids: An overview, Sustainable Cities and Society 51 (2019) 101754.

[25] W. Ye, J. Heidemann, D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, IEEE/ACM Transactions on Networking 12 (3) (2004) 493–506.

[26] T. van Dam, K. Langendoen, An adaptive energy-efficient MAC protocol for wireless sensor networks, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, 2003, pp. 171–180.

[27] G. Lu, B. Krishnamachari, C. Raghavendra, An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks, in: 18th International Parallel and Distributed Processing Symposium, 2004, pp. 224–232.

[28] C. Schurgers, V. Tsiatsis, S. Ganeriwal, M. Srivastava, Optimizing sensor networks in the energy-latency-density design space, IEEE Transactions on Mobile Computing 1 (1) (2002) 70–80.

[29] J. Polastre, J. Hill, D. Culler, Versatile low power media access for wireless sensor networks, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, 2004, pp. 95–107.

[30] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, I. Stoica, A unifying link abstraction for wireless sensor networks, in: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, 2005, pp. 76–89.

[31] X. Wang, L. T. Yang, H. Li, M. Lin, J. Han, B. O. Apduhan, Nqa: A nested anti-collision algorithm for rfid systems, ACM Transactions on Embedded Computing Systems 18 (4).

[32] Telosb, http://openwsn.berkeley.edu/wiki/TelosB.

[33] Tinyos, http://www.tinyos.net/.

[34] C. Merlin, W. Heinzelman, Duty cycle control for low-power-listening MAC protocols, IEEE Transactions on Mobile Computing 9 (11) (2010) 1508–1521.

[35] C. Shen, S. Chen, A cyber-physical design for indoor temperature monitoring using wireless sensor networks, in: 2017 IEEE Wireless Communications and Networking Conference (WCNC), 2017, pp. 1–6.