A Novel Caching Mechanism for Internet of Things (IoT) Sensing Service with Energy Harvesting

Dusit Niyato[1], Dong In Kim[2], Ping Wang[1], and Lingyang Song[3]
[1] School of Computer Engineering, Nanyang Technological University (NTU), Singapore
[2] School of Information and Communication Engineering, Sungkyunkwan University (SKKU), Korea
[3] School of Electrical Engineering and Computer Science, Peking University, Beijing, China

***Abstract*—Caching has shown the success in performance improvement for many wireless communications and networking systems. In this paper, we introduce a caching mechanism for the energy harvesting based Internet of Things (IoT) sensing service. In the service, a sensor harvests energy from an environment. The energy is stored in the battery, and the sensor uses it for sensing and transmitting the reading to the user. A sensing cache can be implemented at a wireless gateway of the sensor to avoid activating the sensor too frequently, hence reducing its energy consumption. We develop an analytical model to investigate the benefit of the proposed caching mechanism. We also introduce the threshold adaptation algorithm that allows the sensing cache dynamically to adjust the parameter of caching to maximize the combined hit rate of the sensing service from multiple sensors. The performance evaluation clearly shows the tradeoff between energy consumption and caching..**

***Index Terms*—Internet of things (IoT), caching, Markov chain**

I. Introduction

Internet of Things (IoT) has been developed as a new paradigm to connect objects, devices, applications, and users without or with minimal human intervention. One of the most common applications of IoT is the sensing service that allows systems and users to monitor environment states, for example, temperature and air pollution level, using a variety of sensors [1]. However, sensors may not have a fixed power supply, and rely on energy harvesting devices. Thus, energy becomes a scarce and crucial resource. A number of approaches have been proposed to reduce energy consumption of sensor devices and improve energy efficiency of sensor networks, e.g., optimized protocols [2]. One of the solutions is caching. Caching allows a fixed facility, e.g., a wireless access point acting as a gateway to a sensor, to store and cache sensing data temporarily. Instead of activating the sensors for every sensing query which consumes considerable energy, the gateway can retrieve the cached sensing data and send it to the user.

Caching is used in IoT services. A cache can be employed at Internet routers [3] to reduce network traffic load. The authors in [4] introduce in-network caching for IoT services. Again, a cache is attached to the content routers in the Internet. The data is tagged with the time and location parameters that help user to identify the most suitable cache to access. The delay due to multihop transmission and cache latency are taken into account to analyze caching performances. With a similar concept, the authors in [5] adopt caching in Named Data Networking (NDN) that allows user interest and data to be matched so as to transfer named content accordingly. The multihop network is considered to deploy caching to reduce energy consumption of the mobile nodes. In [6], caching is used in the IoT Information Centric Networking (ICN) that shows to achieve not only reduced energy consumption, but also less bandwidth usage. Caching is shown to reduce service loss for IoT services in [7] due to multiple copies of data. When applying to the publish-subscribe IoT services, caching can improve the search latency, which is an important performance metric for delay-sensitive traffic [8]. The authors in [9] introduce the cache for a lightweight RFID mutual authentication protocol. The cache is to store a recent visited key of tags that helps to improve the efficiency of authentication with a reader. Caching is applied to IoT services with wireless cooperative transmission in [10]. The study shows that there is an optimal ratio of cached data between the base station and relay so that the total cost of content delivery is minimized. However, there is no existing work that analyzes the impact of caching with the energy harvesting based IoT sensing service. Apart from IoT, caching is used in wireless small cell networks with energy harvesting in [11]. However, the effect on the sensing performance is ignored.

This paper therefore focuses on implementing a caching mechanism to support the energy harvesting based IoT sensing service. We foresee that with limited and random energy supply for the IoT sensor, caching can help to reduce the number of requests sent to the sensor and hence lower its energy consumption. We consider a simple caching mechanism that the sensing data is stored in the cache. The cache is deployed at a fixed wireless access point acting as a gateway for sensors. A timer threshold is set so that if the current sensing data in the cache is still fresh enough, the cache will not obtain sensing result from the sensor but send the cached data to the user. Since the IoT sensing service may combine sensing results from multiple sensors, to achieve an optimal sensing performance, a threshold adaptation algorithm is proposed based on learning algorithm. We additionally introduce an analytical model to investigate the performances of the proposed caching mechanism for the energy harvesting based IoT sensing service. The analysis shows that caching can improve the sensing performance and there is an optimal timer threshold such that the hit rate of the sensing data retrieval is maximized.

II. System Model

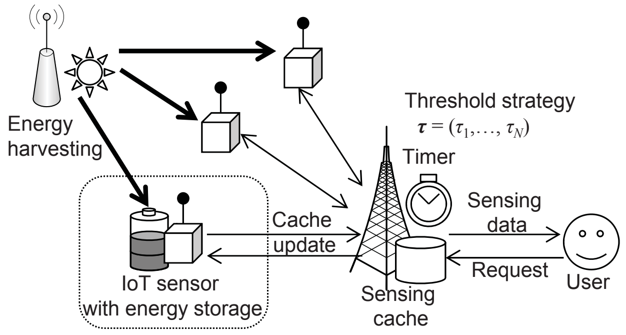


Fig. 1. System model.

We consider an IoT sensing service as shown in Fig. 1. The sensor performs a sensing task, e.g., noise level measurement, from the environment. The sensing result takes a value from a finite set denoted by $\mathcal{S} = \{s_1, \ldots, s_S\}$. The sensor operates on the energy harvested from energy sources, e.g., solar or RF. The probability of successfully harvesting one unit of energy per time unit, i.e., a time slot, is denoted by λ. The harvested energy is stored in a battery with the maximum capacity of B units of energy. We assume that there is a wireless access point acting as a gateway to the sensor. A cache is employed at the access point with a fixed power supply. A user sends a request/query for sensing data to the cache. A request arrives in a time slot with the probability denoted by α. Upon receiving a request, the cache can obtain the sensing result from the sensor. In doing so, the sensor consumes one unit of energy from its battery to collect and transmit sensing data to the cache, which will be forwarded to the user afterward. The cache has a memory to store the sensing result. When the next request arrives, the cache decides whether to retrieve the cached sensing data result stored in its memory or to obtain the fresh sensing data from the sensor for sending to the user.

We assume that the cache adopts a simple threshold based mechanism to update the cached data. The cache maintains a timer t, which represents the freshness of the cached data in its memory. The timer t increases by one for every time slot. If a request arrives, the cache compares the timer with the threshold denoted by τ. If the timer is larger than the threshold, the cache assumes that the cached data is outdated and it activates the sensor and obtains fresh sensing result. Additionally, the cached data is updated, and the timer is reset to zero. Otherwise, the cache sends its cached data to the user. In Section III, we present a model to analyze the performance of the proposed caching mechanism. The sensing service can combine sensing results from multiple sensors to meet the objective and/or achieve better quality of the sensing service. Different sensors can applied different values of threshold. This case will be considered in Section IV, and the threshold adaptation algorithm for multiple sensors will be also presented.

III. Performance Analysis of Single Sensor

The analysis of the caching mechanism for IoT sensing service with energy harvesting is based on a discrete time Markov chain. The system state is observed at the end of time slot. The state space of the Markov chain is defined as follows:

$$\Omega = \Big\{ \big(\mathscr{C}, \mathscr{S}, \mathscr{B}, \mathscr{T}\big) | \mathscr{C}, \mathscr{S} \in \mathcal{S}, \mathscr{B} \in \{0, 1, \ldots, B\}, \mathscr{T} \in \{0, 1, \ldots, T\} \Big\}, \quad (1)$$

where $\mathscr{C}$ and $\mathscr{S}$ are the cached data (state) and actual state of the environment, respectively. $\mathscr{B}$ and $\mathscr{T}$ are the energy level of the sensor's battery and the timer of the cache, respectively. B and T are the maximum capacity of the battery and the maximum value of timer, respectively.

Next, we derive the transition probability matrix of the Markov chain. First, we consider the cached state $\mathscr{C}$ and actual state $\mathscr{S}$. Let $\mathbf{A}$ denote the transition probability of the actual state. Its element is denoted by $A_{s,s'}$ which is the probability that actual state changes from s to s'. There are two cases for the cached state.

- There is no change to the cached state. The resulting transition matrix is obtained from $\mathbf{C}_0 = \mathbf{A} \otimes \mathbf{I}$, where $\otimes$ is the Kronecker product and $\mathbf{I}$ is the identity matrix, which in this case has the same size as that of $\mathbf{A}$.
- The cached state is updated to be the actual state, i.e., when the cache obtains the sensing result from the sensor. Here, let $\mathbf{E}_j$ denote the matrix of zeros except column j which has its values to be ones. The transition matrix in this case is obtained from

$$\mathbf{C}_1 = \begin{bmatrix} A_{1,1}\mathbf{E}_1 & \cdots & A_{1,S}\mathbf{E}_1 \\ \vdots & \ddots & \vdots \\ A_{S,1}\mathbf{E}_S & \cdots & A_{S,S}\mathbf{E}_S \end{bmatrix}. \quad (2)$$

In this case, the cached state will change to the actual state irrespective of its cached data.

Then, we incorporate the transition of the energy level of the sensor. There are two cases.

- The cache is not updated, i.e., the timer is less than or equal to the threshold. Thus, the energy level of the sensor remains the same or increases when there is no or there is energy harvested, respectively. The transition matrix is as follows:

$$\mathbf{B} = \begin{bmatrix} \lambda'\mathbf{C}_0 & \lambda\mathbf{C}_0 & & \\ & \lambda'\mathbf{C}_0 & \lambda\mathbf{C}_0 & \\ & & \ddots & \ddots \\ & & \lambda'\mathbf{C}_0 & \lambda\mathbf{C}_0 \\ & & & 1 \end{bmatrix}, \quad (3)$$

where $\lambda' = 1 - \lambda$.

- The cache can be updated, i.e., the timer is larger than the threshold. Thus, the energy level of the sensor can remain the same, decrease, or increase, depending on whether there is an arriving request and whether energy is harvested or not. We consider two subcases of having and not having an arriving request. For the former, the energy level can decrease, and the transition matrix is

$$\mathbf{B}_0 = \begin{bmatrix} \mathbf{0} & & & & \\ \alpha\lambda'\mathbf{C}_1 & \alpha\lambda\mathbf{C}_1 & & & \\ & \ddots & \ddots & & \\ & & \alpha\lambda'\mathbf{C}_1 & \alpha\lambda\mathbf{C}_1 & \end{bmatrix}, \quad (4)$$

where $\mathbf{0}$ is a matrix of zeros with an appropriate size. For the latter, the energy level can increase, and the transition matrix is

$$\mathbf{B}_1 = \begin{bmatrix} \lambda'\mathbf{C}_0 & \lambda\mathbf{C}_0 & & & \\ & \alpha'\lambda'\mathbf{C}_0 & \alpha'\lambda\mathbf{C}_0 & & \\ & & \ddots & \ddots & \\ & & & \alpha'\lambda'\mathbf{C}_0 & \alpha'\lambda\mathbf{C}_0 \\ & & & & (\alpha'\lambda' + \alpha'\lambda)\mathbf{C}_0 \end{bmatrix}. \quad (5)$$

where $\alpha' = 1 = \alpha$. Finally, we combine the transition of the timer. The transition matrix is expressed as follows:

$$\hat{\mathbf{P}} = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & & & \\ \mathbf{P}_{1,0} & \mathbf{0} & \mathbf{P}_{1,2} & & \\ \vdots & & & \ddots & \\ \mathbf{P}_{T-1,0} & & & & \mathbf{P}_{T-1,T} \\ \mathbf{P}_{T,0} & & & & \mathbf{P}_{T,T} \end{bmatrix}, \quad (6)$$

where $\mathbf{P}_{t,t'}$ is the transition matrix for the timer changing from t to t'. Each row of this matrix is from the corresponding row of matrices $\mathbf{B}$, $\mathbf{B}_0$, and $\mathbf{B}_1$ depending on the timer threshold τ. Let $[\mathbf{P}_{t,t'}]_{(c,s,b)}$ denote the row of matrix $\mathbf{P}_{t,t'}$ corresponding to cached state c, actual state s, and energy level b, and similarly for matrices $\mathbf{B}$, $\mathbf{B}_0$, and $\mathbf{B}_1$. We have

$$[\mathbf{P}_{t,0}]_{(c,s,b)} = [\mathbf{B}_0]_{(c,s,b)}, \quad t = 0, 1, \dots, T, \quad (7)$$

for $t > \tau(c)$. Here, the threshold $\tau(c)$ is defined as a function of cached state c. In this case, there is an arriving request, and consequently the timer is reset to zero.

$$[\mathbf{P}_{t,t+1}]_{(c,s,b)} = [\mathbf{B}_1]_{(c,s,b)}, \quad t = 0, 1, \dots, T-1, \quad (8)$$

for $t > \tau(c)$. In this case, there is no arriving request, and consequently the timer increases by one. However, if the timer reaches the maximum value T, the timer remains the same, i.e.,

$$[\mathbf{P}_{T,T}]_{(c,s,b)} = [\mathbf{B}_1]_{(c,s,b)}. \quad (9)$$

For $t \le \tau(c)$, we have

$$[\mathbf{P}_{t,t+1}]_{(c,s,b)} = [\mathbf{B}]_{(c,s,b)}, t = 0, 1, \dots, T-1. \quad (10)$$

In this case, the timer increases regardless of whether there is an arriving request or not. The cached state is sent to the user.

The steady state probability of the Markov chain is defined as $\pi(c, s, b, t)$ for the cached state c, actual state s, energy level b, and timer t. Let $\vec{\boldsymbol{\pi}}$ be the vector of the steady state probability, i.e., its element is $\pi(c, s, b, t)$. The vector can be obtained from solving $\vec{\boldsymbol{\pi}}^\top \hat{\mathbf{P}} = \vec{\boldsymbol{\pi}}^\top$ and $\vec{\boldsymbol{\pi}}^\top \vec{\mathbf{1}} = 1$, where $\vec{\mathbf{1}}$ is a vector of ones with an appropriate size and $\hat{\mathbf{P}}$ is the transition probability matrix obtained from (6).

The hit rate of the IoT sensing service is defined as the probability that the user receives the true state of the environment. This happens when (I) the cached state is the same as the actual state if the cache decides not to obtain the sensing result from the sensor, or (II) the cache requests for the sensing result and the sensor has enough energy, or (III) if the cache requests for the sensing result but the sensor does not have enough energy, the cached state must be the same as the actual state. The probability is then obtained from

$$\begin{aligned} P_{\text{hit}} = & \underbrace{\sum_{c=s\in\mathcal{S}} \sum_{b=0}^{B} \sum_{t=0}^{\tau(c)} \pi(c, s, b, t)}_{\text{(I)}} \\ & + \underbrace{\sum_{c,s\in\mathcal{S}} \sum_{b=1}^{B} \sum_{t=\tau(c)+1}^{T} \pi(c, s, b, t)}_{\text{(II)}} \\ & + \underbrace{\sum_{c=s\in\mathcal{S}} \sum_{b=0} \sum_{t=\tau(c)+1}^{T} \pi(c, s, b, t)}_{\text{(III)}}. \end{aligned} \quad (11)$$

IV. Threshold Adaptation

In this section, we consider the cache serving multiple sensors. The sensing results from these sensors are combined by a central sensing cache. For example, spectrum sensors can be deployed and their results are aggregated to obtain a final sensing outcome. The major challenge is to choose an optimal threshold strategy such that the sensing hit rate of the service is maximized. We propose the threshold adaptation algorithm to let the sensing cache learn and achieve an optimal threshold strategy.

Let $\boldsymbol{\tau} = (\tau_1, \dots, \tau_N)$ denote the joint threshold strategy for N sensors of the service, and Ω denote the set of all possible strategies, i.e., $\boldsymbol{\tau} \in \Omega$. Let $P_{\text{hit}}(n, \tau_n)$ denote the hit probability of sensor n given threshold τ_n obtained from (11). The objective of the sensing service is to maximize the combined hit rate of all sensors, i.e.,

$$\max_{\boldsymbol{\tau}\in\Omega} \prod_{n=1}^{N} P_{\text{hit}}(n, \tau_n). \quad (12)$$

This combined hit rate or success rate represents the fact that the accuracy of the final sensing result depends on all individual sensing results, e.g., using AND operation. To obtain an optimal joint threshold strategy, the sensing cache can adopt a learning algorithm as shown in Algorithm 1. First, the cache initializes current threshold τ_n. Given the current thresholds, the cache randomly chooses one of the sensors to possibly switch to new threshold τ_n'. The cache evaluates if the final sensing result is better or not. $rand()$ is a random number generator function. Here, with small probability ϵ, i.e., the probability of random threshold adjustment, the cache can switch to the new threshold irrationally. This is the exploration for the better threshold.

Algorithm 1 Threshold adaptation algorithm

```
1: Initialize $\tau_n = 1$. Initialize estimated hit probability
   $H_n(\tau_n) \leftarrow 0$
2: loop
3:   Measure hit probability of all sensors $H_n(\tau_n)$
4:   Randomly choose one of sensors to adapt its threshold
     and select new threshold $\tau_n'$
5:   if $\prod_{n=1}^{N} H_n(\tau_n') > \prod_{n=1}^{N} H_n(\tau_n)$ then
6:     Switch to new threshold $\tau_n \leftarrow \tau_n'$
7:   else
8:     if $rand() < \epsilon$ then
9:       Switch to new threshold $\tau_n \leftarrow \tau_n'$
10:    end if
11:  end if
12: end loop
```

We formulate a Markov chain to analyze the performance of the threshold adaptation algorithm. The state is the thresholds of all sensors, i.e., $\boldsymbol{\tau} = (\tau_1, \dots, \tau_N)$, where $\tau_n \in \{1, \dots, \tau_{\max}\}$ and $\tau_{\max}$ is the largest possible threshold. Let $q_{\boldsymbol{\tau},\boldsymbol{\tau}'}$ denote the transition rate from current state $\boldsymbol{\tau}$ to next state $\boldsymbol{\tau}'$. The transition rate can be expressed as follows:

$$q_{\boldsymbol{\tau},\boldsymbol{\tau}'} = \begin{cases} \frac{1}{N\tau_{\max}}, & \text{if } \prod_{n=1}^{N} P_{\text{hit}}(n,\tau_n') > \prod_{n=1}^{N} P_{\text{hit}}(n,\tau_n), \\ \epsilon, & \text{if } \prod_{n=1}^{N} P_{\text{hit}}(n,\tau_n') \leq \prod_{n=1}^{N} P_{\text{hit}}(n,\tau_n). \end{cases} \tag{13}$$

Here, the transition rate of the self-transit state is given by

$$q_{\boldsymbol{\tau},\boldsymbol{\tau}} = -\sum_{\boldsymbol{\tau}' \neq \boldsymbol{\tau}} q_{\boldsymbol{\tau},\boldsymbol{\tau}'}. \tag{14}$$

The transition rate matrix can be expressed as in (15). It is straightforward to show that this Markov chain is ergodic and irreducible. Then, the steady state probability of the threshold strategy reached by the threshold adaptation algorithm can be obtained by solving the following equations:

$$\vec{\mathbf{p}}^\top \mathbf{Q} = \vec{\mathbf{0}}^\top, \quad \vec{\mathbf{p}}^\top \vec{\mathbf{1}} = 1, \tag{16}$$

where $\vec{\mathbf{0}}$ and $\vec{\mathbf{1}}$ are a vector of zeros and a vector of ones, respectively. $\vec{\mathbf{p}}$ is a vector containing steady state probability $p_{(\tau_1,\dots,\tau_N)}$ of threshold strategy $\tau_1, \dots, \tau_N$, i.e.,

$$\vec{\mathbf{p}} = \begin{bmatrix} p_{(1,\dots,1)} \\ \vdots \\ p_{(\tau_1,\dots,\tau_N)} \\ \vdots \\ p_{(\tau_{\max},\dots,\tau_{\max})} \end{bmatrix}. \tag{17}$$

For the steady state probability $1 \geq p_{(\tau_1,\dots,\tau_N)} \gg 0$, its corresponding state $(\tau_1, \dots, \tau_N)$ can be a candidate for an optimal threshold strategy. When the probability of random threshold adjustment ϵ becomes small, the transition that results in lower combined hit rate will diminish. Thus, the optimal threshold strategy will become an absorbing state of the Markov chain. Let $\boldsymbol{\tau}^* = (\tau_1^*, \dots, \tau_N^*)$ denote an optimal threshold strategy of the sensing cache. We have $q_{(\tau_1^*,\dots,\tau_N^*),(\tau_1,\dots,\tau_N)} \to 0$ for $(\tau_1, \dots, \tau_N) \neq (\tau_1^*, \dots, \tau_N^*)$ when $\epsilon \to 0$.

Given that $\epsilon = 0$, we can give a canonical form of the absorbing Markov chain as follows:

$$\hat{\mathbf{Q}} = \left[\begin{array}{c|c} \mathbf{H} & \vec{\mathbf{h}} \\ \hline \mathbf{0}^\top & 0 \end{array}\right], \tag{18}$$

where $\mathbf{H}$ is the state transition matrix among transient states, i.e., for $\boldsymbol{\tau}' \in \Omega \backslash \{(\tau_1^*, \dots, \tau_N^*)\}$, and $\vec{\mathbf{h}}$ is the vector containing the transition rates to the absorbing state. Here $\left[\begin{array}{c|c} \mathbf{0}^\top & 0 \end{array}\right]$ indicates that from the absorbing state the transition to any transient states is not possible. Let the algorithm start from the threshold equal to one for all sensors, the initial transient state probability is denoted by $\boldsymbol{\alpha} = \left[\begin{array}{cccc} 1 & 0 & \cdots & 0 \end{array}\right]$. Then, the average time to reach the optimal threshold strategy is given by

$$\theta = -\boldsymbol{\alpha}\mathbf{H}^{-1}\vec{\mathbf{1}}, \tag{19}$$

which is the average time to absorption of an absorbing Markov chain.

V. Performance Evaluation

A. Parameter Setting

The sensor has a battery with the capacity of 10 units of energy. The sensor can harvest one unit of energy with probability 0.2. Unless otherwise stated, the maximum timer of the cache is 10. The environment has 4 states, the set of which is $\mathcal{S} = \{1, 2, 3, 4\}$, and their probabilities are uniform. The sojourn time of each state is 10 time slots.

B. Numerical Results

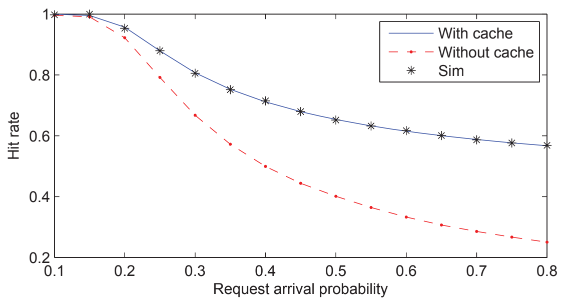


Fig. 2. Hit rate under varied request arrival probability.

1) Single Sensor: Figure 2 shows the comparison of a hit rate between the energy harvesting based IoT sensing service with and without a cache. Without a cache, the arriving request

$$\mathbf{Q} = \begin{bmatrix} q_{(1,\ldots,1),(1,\ldots,1)} & \cdots & q_{(1,\ldots,1),(\tau_1,\ldots,\tau_N)} & \cdots & q_{(1,\ldots,1),(\tau_{\max},\ldots,\tau_{\max})} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q_{(\tau_1,\ldots,\tau_N),(1,\ldots,1)} & \cdots & q_{(\tau_1,\ldots,\tau_N),(\tau_1,\ldots,\tau_N)} & \cdots & q_{(\tau_1,\ldots,\tau_N),(\tau_{\max},\ldots,\tau_{\max})} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q_{(\tau_{\max},\ldots,\tau_{\max}),(1,\ldots,1)} & \cdots & q_{(\tau_{\max},\ldots,\tau_{\max}),(\tau_1,\ldots,\tau_N)} & \cdots & q_{(\tau_{\max},\ldots,\tau_{\max}),(\tau_{\max},\ldots,\tau_{\max})} \end{bmatrix} \quad (15)$$

will be rejected if the sensor does not have enough energy to perform sensing and transmitting the result to the user. With a cache, the timer threshold is set to 2 for all cached states. From Fig. 2, clearly the hit rates of the sensing service with and without cache are high and indistinguishable when the request arrival probability is small, e.g., 0.1. However, when the request arrival probability increases, having a cache can improve the hit rate significantly. This is because obtaining the sensing result consumes energy of the sensor. With too many requests, the battery is frequently depleted and the sensor cannot transmit the sensing result to the cache and user. Caching can reduce energy consumption of the sensor. The simulation is conducted to validate the correctness of the analysis. The analytical results and simulation results match well in our evaluation.

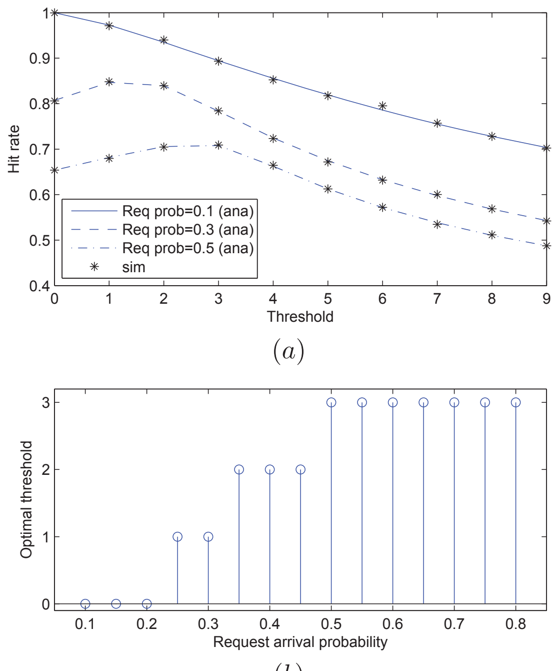


Fig. 3. (a) Hit rate under different thresholds and (b) optimal threshold.

Next, we vary the timer threshold to update the cache. With a different request arrival probability, the hit rates are varied (Fig. 3(a)). When the request arrival probability is small, e.g., 0.1, the sensor has enough opportunity to harvest energy and is able to transmit the correct state to the user. Having a cache is not important here. However, at higher request arrival probabilities, e.g., 0.3 and 0.5, the cache should not be updated too frequently as that will exhaust the energy harvested at the sensor. In this case, the timer threshold of the sensor can be set such that the maximum hit rate can be achieved. Figure 3(b) shows the optimal timer threshold. With the same aforementioned reason, the optimal threshold increases as the request arrival probability increases.

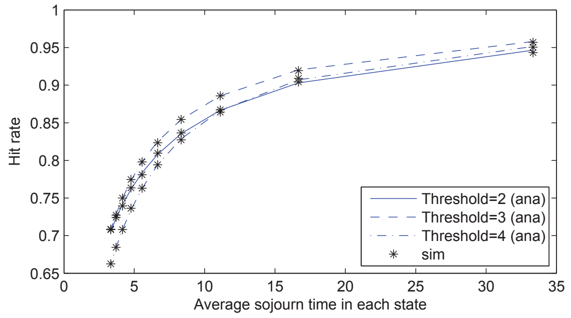
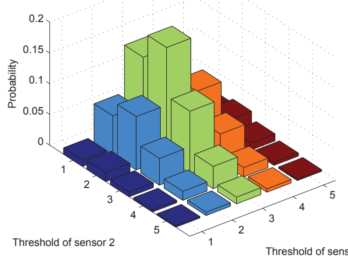


Fig. 4. Hit rate under varied sojourn time at each state.

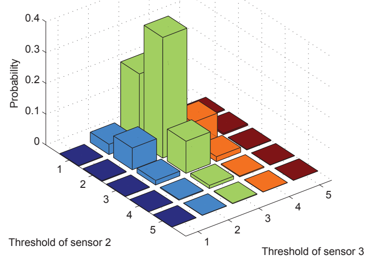
Figure 4 shows the hit rate when the sojourn time of the environment at each state is varied. We consider the high request arrival probability, i.e., 0.5. Note that the probability of each state is still uniform in this case. When the sojourn time increases, the environment remains at the same state longer. Thus, having cache becomes more effective and improves the performance significantly. Again, we observe that the timer threshold of 3 is optimal to achieve the highest hit rate.

2) Multiple Sensors: Next, we consider the threshold adaptation for multiple sensors. There are four sensors 1, 2, 3, and 4, and their request probabilities are 0.2, 0.3, 0.4, and 0.5, respectively. The probability of random threshold adjustment is varied with $\epsilon = 10^{-3}$, $\epsilon = 10^{-4}$, and $\epsilon = 10^{-6}$ for Figs. 5(a), (b), and (c), respectively. In these figures, we show the thresholds for sensors 2 and 3, while those of sensors 1 and 4 are averaged. Clearly, when the probability of random threshold adjustment is large, there is much fluctuation of certain thresholds to be selected for the sensors (e.g., Fig. 5(a)). When the probability becomes small, the threshold strategy which is optimal is likely to be selected (e.g., Fig. 5(c)). In this case, the optimal thresholds for sensors 2 and 3 are $\tau = 2$ and $\tau = 3$, respectively, while those of sensors 1 and 4 are $\tau = 1$ and $\tau = 4$ as shown in Fig. 4(b).

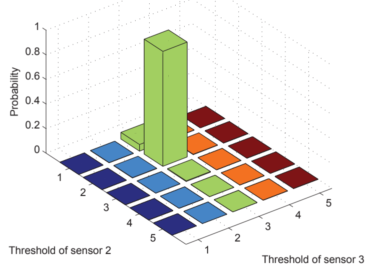
With $\epsilon = 10^{-6}$, Fig. 6 shows the threshold strategy convergence. From the learning algorithm, the sensing cache is able to adjust dynamically the threshold strategy of sensors used in the IoT services. Note here that there is a small



(a)



(b)



(c)

Fig. 5. Probabilities of taking different strategy of threshold adaptation for (a) $\epsilon = 10^{-3}$, (b) $\epsilon = 10^{-4}$, and (c) $\epsilon = 10^{-6}$.

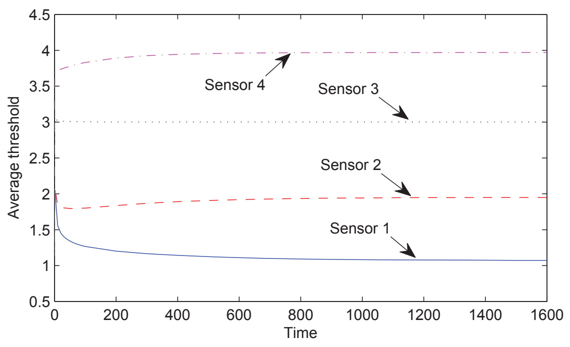


Fig. 6. Convergence of threshold adaptation.

discrepancy between learned thresholds and the optimal ones. This is from the fact that the probability of random threshold adjustment contributes to a slight fluctuation of the strategy, which is required to let the sensing cache evaluate other possible threshold strategies, i.e., exploration.

VI. Summary

We have introduced the use of cache for Internet of Things (IoT) sensing service with energy harvesting. The cache can be located at a wireless gateway that provides a connection to a sensor. We have proposed a simple caching mechanism that allows the sensing result to be stored in the cache for a certain time period, and the user can retrieve the sensing result from the cache. For the sensor with energy harvesting, the cache can reduce energy consumption of the sensor and thus improve the sensing success performance. In addition to the caching mechanism, we have presented an analytical model of the IoT sensing service with caching. The numerical studies clearly show the benefit of the cache used in the IoT sensing services. The analytical model can be used to determine an optimal time period that the cache should be updated.

For the future work, we will consider the sensing error and employ the advanced cache updating algorithm for multiple sensors.

Acknowledgements

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (2014R1A5A1011478) and Singapore MOE Tier 1 (RG18/13 and RG33/12) and MOE Tier 2 (MOE2014-T2-2-015 ARC 4/15).

References

[1] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787-2805, October 2010.

[2] S. Sudevalayam and P. Kulkarni, “Energy harvesting sensor nodes: Survey and implications,” *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 443-461, Third Quarter 2011.

[3] J. Li, Y. Shvartzshnaider, J. Francisco, R. P. Martin, and D. Raychaudhuri, “Enabling Internet-of-Things services in the MobilityFirst Future Internet Architecture,” in *Proceedings of IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2012.

[4] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, “In-network caching of Internet-of-Things data,” in *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 3185-3190, June 2014.

[5] M. A. Hail, M. Marica, A. Molinaro, and S. Fischer, “Caching in named data networking for the wireless Internet of Things,” in *Proceedings of International Conference on Recent Advances in Internet of Things (RIoT)*, April 2015.

[6] J. Quevedo, D. Corujo, and R. Aguiar, “A case for ICN usage in IoT environments,” in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, pp. 2770-2775, December 2014.

[7] F. Ganz, R. Li, P. Barnaghi, and H. Harai, “A resource mobility scheme for service-continuity in the Internet of Things,” in *Proceedings of IEEE International Conference on Green Computing and Communications (GreenCom)*, pp. 261-264, November 2012.

[8] Y. Sun, X. Qiao, B. Cheng, and J. Chen, “A low-delay, lightweight publish/subscribe architecture for delay-sensitive IOT services,” in *Proceedings of IEEE International Conference on Web Services (ICWS)*, pp. 179-186, June - July 2013.

[9] K. Fan, C. Liang, H. Li, and Y. Yang, “LRMAPC: A lightweight RFID mutual authentication protocol with cache in the reader for IoT,” in *Proceedings of IEEE International Conference on Computer and Information Technology (CIT)*, pp. 276-280, September 2014.

[10] Y. Nam, J.-H. Park, and J.-M. Chung, “Performance analysis of cooperative content delivery in wireless IoT networks,” in *Proceedings of IEEE International Symposium on Consumer Electronics (ISCE)*, June 2014.

[11] S. Zhou, J. Gong, Z. Zhou, W. Chen, and Z. Niu, “GreenDelivery: proactive content caching and push with energy-harvesting-based small cells,” *IEEE Communications Magazine*, vol. 53, no. 4, pp. 142-149, April 2015.