Signal Processing 126 (2016) 77–86

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Estimator Goore Game based quality of service control with incomplete information for wireless sensor networks

Shenghong Li [a,*], Hao Ge [a], Ying-Chang Liang [b], Feng Zhao [c], Jianhua Li [a]

[a] Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[b] Institute for Infocomm Research, A*STAR, Singapore
[c] Gunlin University of Electronic Technology, Guangxi, China

ARTICLE INFO

Article history:
Received 25 July 2015
Received in revised form
16 November 2015
Accepted 22 November 2015
Available online 11 December 2015

Keywords:
Wireless sensor networks
Quality of Service
Goore game
Incomplete information

ABSTRACT

Quality of Service (QoS) control is one of the key mechanisms to ensure the effective operation of wireless sensor networks (WSNs). In many specific applications such as battlefield communications, the optimum number of sensors sending information at any given time is an important QoS requirement. For such requirement, some Goore Game based QoS control approaches have been exploited to dynamically adjust the number of active sensors to the optimal one in the literatures, but they assume that the optimum number is known to the base station explicitly, and each method has its own shortcomings.

In this paper, a novel approach is proposed to solve the QoS control problem. A framework for solving such problem is firstly proposed, in which the optimum sensor number is not known in advance but learnt by interacting with upper level applications in a reinforcement manner. On this basis, an algorithm named Estimator Goore Game (EGG) is designed to adjust the number of active sensors effectively. The proposed algorithm cannot only relax the above assumption, but also has a competitive performance. Furthermore, the algorithm is also applicable for both stationary and non-stationary environments. Simulation results on the convergence time, the network life and the ability of tracking dynamic QoS have demonstrated the effectiveness of the proposed algorithm.

1. Introduction

Wireless sensor networks (WSNs) are designed to gather information about the state of physical world and transmit sensed data to interested users, and have been widely applied to many scenarios such as health care monitoring [1,2], battlefield communication [3], space exploration [4,5], and event detection [6]. While WSNs have a broad prospect of applications, due to the limitation of each node's computing ability, communication ability, caching ability and power supply, transmission of data in such applications entails both energy and Quality of Service (QoS) support in order to ensure efficient usage of the sensor resources and effective access to the gathered information [7]. Over the past decades, some important aspects of WSNs such as architecture and protocol design, energy conservation, and locationing have been extensively studied [8]. However, little attention has been paid to supporting QoS and there is currently no standardization on framework or general guidelines in the networking community on how QoS can be achieved in WSNs [9]. With the emergence of more and more critical, multimedia and real-time applications, QoS control is becoming an emergent issue that should be addressed.

* Corresponding author.
E-mail addresses: shli@sjtu.edu.cn (S. Li), sjtu_gehao@sjtu.edu.cn (H. Ge), ycliang@i2r.a-star.edu.sg (Y.-C. Liang), zhaofeng@guet.edu.cn (F. Zhao), lijh888@sjtu.edu.cn (J. Li).

http://dx.doi.org/10.1016/j.sigpro.2015.11.021
0165-1684/

It is well known that QoS is an overused term with various meanings and different technical communities may interpret QoS in different ways [10,8]. Conceptually, the term QoS can be regarded as the capability to provide assurance that the service requirements of applications can be satisfied [11]. It is intuitive that different applications may have different QoS requirements. In the case of many specific applications such as battlefield communications and space exploration, due to sensor death or replenishment, the number of active sensors is an important QoS requirement parameter.

For such situations, Iyer and Kleinrock [12] suggest defining the QoS of sensor networks as the optimum number of sensors that should be sending information to a base station (or sink node) at any given time. Based on this QoS definition, under the assumption that the optimum number of the active sensors is known to base station in advance, [12] proposes a QoS control approach, in which the base station communicates QoS information to each of the sensors by using a broadcast channel and the mathematical paradigm of the Goore Game is exploited to dynamically adjust the number of active sensors to the optimum. However, the original Goore Game approach in [12] is not power-aware, and once the WSN reaches the optimum number of active sensor nodes in the system, those active sensor nodes will remain active for every subsequent epoch until their battery power is depleted or the QoS requirement (optimum number of active nodes) is changed. To solve this problem, several improved approaches are presented with the same assumption above [13–17]. Network life is concerned and the definition of *network life* is provided in [13]. Zhao et al. [14] devoted to maximize the lifetime of network by having sensors periodically power-down to conserve their battery energy. The sensors that are in deep sleep states can power off their MCU (Micro-Controller Unit) and radio system, and a timer is utilized to control the time length of the deep sleep. Ayers [15] developed the original Goore Game model to further address the power consumption problem. Three new mechanisms are introduced into the game: player rotation, proactive referee, and unambiguous reward/punishment. In [16], convergence time and convergence range is improved significantly by using flags to exclude redundant sensors from the game. However, the drawbacks are also obvious, when the desired QoS changes with time or sensor nodes deplete, the algorithm appears to be less attractive, and it leads to the imbalance consumption of system energy further. Adaptive Periodic Goore Game is proposed in [17], the Goore Game is reapplied periodically to alleviate the unbalance of energy consumption, and also the idea of unambiguous reward/punishment is borrowed from [15] to accelerate the convergence. But the periodic restart of the game will lead to a periodical instable performance.

Considering that the approaches above assume that the base station knows explicitly how many active sensors will satisfy the QoS requirement of application, which is often unreasonable in many real environments, and that the approaches have their own other limitations, in the paper we propose a new framework for controlling the QoS with the aforementioned definition. In our proposed framework, the optimum number of the active sensors is learnt by a reinforcement manner, i.e., the upper level application will respond to the information collected by the base station and feed back a signal indicating whether the collected information is enough for supporting application or not. In addition, the approach designs an Estimator Goore Game (EGG) algorithm as an implementation of the proposed framework.

Our work presents a set of novel contributions that are summarized as follows:

1. In the existing QoS control approaches mentioned above, it is assumed that the optimum number of active sensors is known by the base station (or sink node) in advance. Instead, in our proposed approach, the optimum number is learnt in a reinforcement way. To the best of our knowledge, we present the first reported approach for the situation where the optimum number is not known explicitly, and that is the reason we call it *incomplete information* in the title.
2. We provide a rigorous analysis on the behavior of the proposed EGG under stationary environments. The EGG is proved to be able to converge to the required QoS within finite time in any stationary environment almost surely.
3. Simulation results show that our proposed approach outperforms all the existing approaches with regard to network life maintenance. For the proposed approach, all alive nodes can be efficiently employed to support QoS requirement.
4. It is rather fascinating to demonstrate that our proposed approach also possesses an excellent ability to cope with non-stationary environments which may be caused by QoS requirement change, sensor node being out of battery, sensor node battery recharging, sensor node redeployment, etc. In such situations, our approach can also exhibit a fast and accurate tracking ability.

The remainder of this paper is organized as follows. In Section 2, we establish the framework of QoS control for WSNs with incomplete information. On this basis, an Estimator Goore Game algorithm is designed and its behavior is analyzed in Section 3. We run simulations and the results are presented in Section 4, various aspects of analysis are also available in that section. Section 5 concludes the paper.

2. System model

In all the aforementioned literatures, it is assumed that the base station (or sink node) knows the optimum number of active sensors explicitly. However, this assumption holds in very limited circumstances. To be more general, we propose a modified framework to solve the QoS control problem.

Within the proposed framework, as shown in Fig. 1, an application-specific WSN system consists of geographically distributed sensors and a base station. Sensors gather information about states of the physical world and

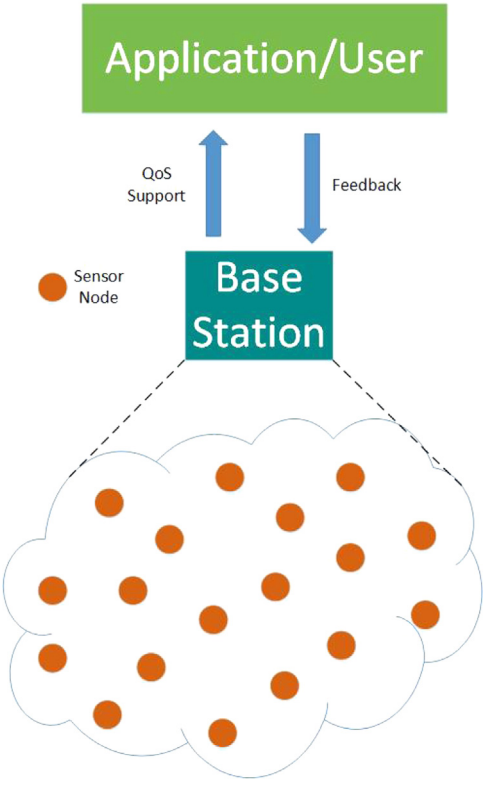


Fig. 1. QoS control model.

communicate with the base station via wireless links. Meanwhile, the base station forwards the information to upper level applications (or users) via wired links. The upper level applications (or users) will judge if the information is enough for QoS requirement or not and give a feedback to the base station. The base station will maintain an estimate for the optimum number of active sensors. Based on the feedback received, the estimate for the optimum number of active sensors is adjusted accordingly. And then the base station will adjust the number of active sensors toward the estimated value through the mathematical paradigm of the Goore Game.

2.1. Goore Game for QoS control in WSN

We describe the original Goore Game algorithm given in [8] briefly as follows. Each sensor node in WSNs acts as one of player roles in the Goore Game, each of whom can decide to vote *yes* or *no* independently in every trial, and the base station (or sink node) acts as the referee, who gives a reward probability based on the number of players that vote *yes* every trial. The reward probability is broadcasted to every player, and then all the players reward themselves independently with the reward probability. A typical reward probability generating function advocated in [8] is

$$r(k_t) = 0.2 + 0.8e^{-0.002(k_t - k^*)^2} \quad (1)$$

where k_t is the number of nodes voting *yes* in trial t, and k^* is the desired number. Fig. 2 illustrates an example of the reward probability generating function with $k^* = 35$. One intriguing property of Goore Game is that there will be approximately k^* *yes* votes after enough trials.

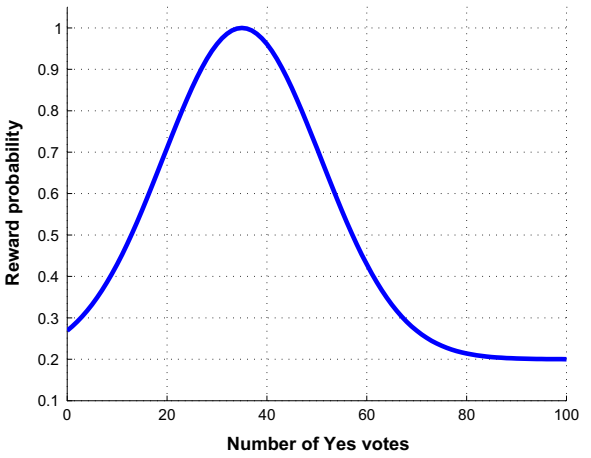


Fig. 2. Example of reward probability generating function with $k^* = 35$.

Here, voting *yes* implies an *active state* of the sensor (it will send information to base stations) and *no* for *stand-by state* of the sensor (temporally power-off to save energy).

The Goore Game approach is only applicable to one-hop WSNs with star-topology, in which every sensor node can communicate with the base station (or sink node). When the scale of a network is large, usually clusters should be effectively organized so that the Goore Game can be applicable to each cluster [15]. And it is assumed that the information is uniformly distributed over all the sensors within such a cluster, therefore the information gathered is in proportion to the number of sensors sending packets to base station.

2.2. Application-base station interaction

As described above, the performance of the original Goore Game depends highly on the design of reward probability generating function, and the typical reward probability generating function given in Eq. (1) requires the exact value of optimum number k^*. However, in realistic applications, k^* is usually unknown to the information gathering end (base station or sink node), the only available information is a feedback given by upper-level application, indicating whether the information gathered is adequate or not. Therefore in our framework, the optimum number k^* is not explicitly known to us, instead, a feedback $f(k) = \text{sign}(k - k^*)$ is given.

Obviously, the feedback $f(k)$ holds the following two properties:

Property 1: *If QoS requirement can be satisfied by activating k sensors, then it definitely will also be achieved by activating k' (larger than k) sensors, i.e., $f(k) = 1 \Rightarrow f(k') = 1$, where $k' \geq k$.*

Property 2: *Vice versa, if k active sensors fail to meet QoS requirement, then any k' (less than k) sensors*

cannot meet QoS requirement, i.e., $f(k)=0 \Rightarrow f(k')=0$, *where* $k' \le k$.

Regarding QoS control problem, on one hand, we must guarantee the requirement of QoS of upper-level application being satisfied, on the other hand, we must reduce the number of active sensors at any time instant as much as possible. That is to say, we are trying to find the minimum k that satisfies $f(k)=1$.

3. QoS control with incomplete information

Based on the framework proposed in the last section, we will propose our algorithm hereinafter.

3.1. Algorithm description

As the preceding framework reveals, the proposed algorithm communicates the information of QoS desired by upper level applications with the sensors, devoting to coordinate the number of active sensors to the optimum. The following Algorithm 1 describes the designed Estimator Goore Game (EGG). In this algorithm, three strategies are involved as follows.

Algorithm 1. Estimator Goore Game.

Require: k^*: optimum number of active sensors; M: total number of active sensors; α: diminishing parameter; β: perturbation parameter; $r(k)$: reward probability generating function; k_t: number of active sensors at epoch t;
Ensure: $k_t \to k^*$
  initial $count=0$;
  loop
    update $\hat{k}^*$ according to Algorithm.2
    if k_t unchanged **then**
      $count++$;
  else
    $count=0$;
  end if
  if $k_t=\hat{k}^*$ **then**
   $RewardProbability=\beta^{count}\times r(k_t)$
  else
    $RewardProbability=\alpha^{count}\times r(k_t)$
  end if
  if $k_t<\hat{k}^*$ **then**
    for all Live Stand-by Sensors **do**
    Update themselves with RewardProbability
    end for
  else
    for all Live Active Sensors **do**
    Update themselves with RewardProbability
   end for
  end if
end loop

3.1.1. Strategy of sensor nodes

As in [12], a fixed structure learning automaton (FSLA), typically a two-action Tsetlin automaton [18], is associated with each sensor node. The two actions depend on the internal state of the automaton, indicating the active or stand-by state of the sensor node. Sensor nodes in stand-by state will only receive signals thus preserving energy. All the sensors will update their own states according to the reward probability received from the base station (or sink node) independently. The automaton's internal state transition is illustrated as Fig. 3, where N is the memory depth of the automaton. $a=0$ and $a=1$ represent respectively that an automaton is in stand-by state and active state. $r=1$ and $r=0$ stand for a reward with probability $r(k_t)$ and a penalty with probability $1-r(k_t)$ respectively. At any epoch t, k_t is the total number of sensors that in state $a=1$, i.e., $k_t=\sum_i a_i$.

3.1.2. Strategy of base station

In the proposed framework, because k^* is not known in advance, here an estimate $\hat{k}^*$ of optimum number k^* is used to generate the reward probability according to $r(k_t)=0.2+0.8e^{-0.002(k_t-\hat{k}^*)^2}$, and then the reward probability is broadcasted to all the sensors in the network. The estimate $\hat{k}^*$ is updated according to Algorithm 2.

Algorithm 2. Estimator updating strategy.

Require: *feedback*: feedback received from upper level application;
  k_t: number of active sensors at epoch t;
Ensure: $\hat{k}^*$
  initial $lb=0$, $ub=\infty$;
  If *feedback*$=0$(QoS unsatisfied) **then**
    If $lb=ub$ **and** $lb<=k_t$ **then**
      $ub=\infty$;
    end if
    $lb=\max(k_t+1, lb)$;
else
  If $lb=ub$ **then**
    If $k_t<lb$ **then**
      $lb=0$;
      $ub=k_t$;
    end if
  else
    $ub=\min(\max(k_t, lb), ub)$;
    end if
end if
If $|k_t-lb|>|k_t-ub|$ **then**
  return $\hat{k}^*=lb$
else
  return $\hat{k}^*=ub$
end if

We may note from Fig. 2 that when k_t approaching k^*, all the sensors will be rewarded with a very high probability. For example, $r(39)=0.9984$ and $r(38)=0.9936$ with $k^*=40$, which will slow the convergence remarkably. In our algorithm, the idea of *diminishing reward probability* is proposed to solve this problem. The algorithm counts the number of consecutive epochs that k_t stay unchanged, denoted as *count*, and the reward probability will be diminished by multiplying by the factor α^{count}, where α is a diminishing parameter. It is obvious that a smaller α represents a larger diminishing rate.

After the number of active sensors reaches the estimated optimum number, all sensors will be rewarded with probability 1, the system does not tend to change its state any longer. In the case of varying k^*, the system is unable to track a changing optimum number. To address this issue, we introduced a perturbation parameter β. Reward probability will be diminished by multiplying by the factor β^{count} to give the system a little perturbation to explore

new possibilities. β can be trivially set as 1 in stable QoS requirement scenarios. A small β is appropriate for use in an environment with a fiercely changing QoS requirement.

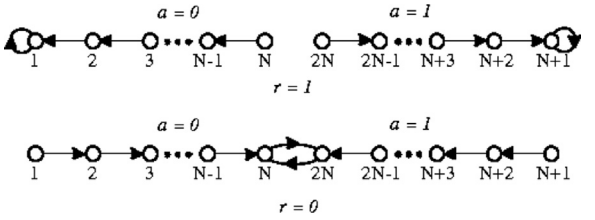


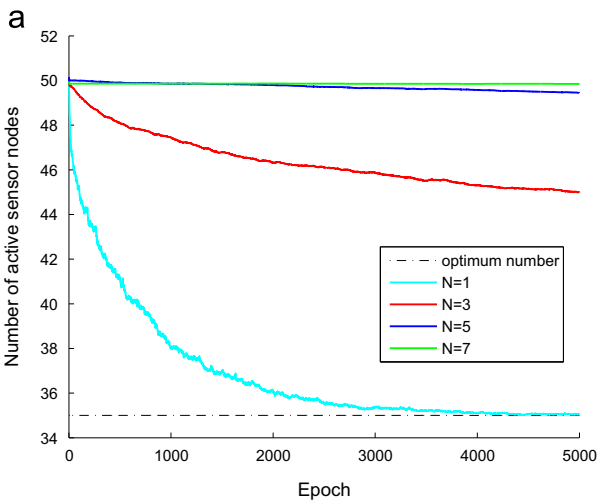
Fig. 3. A Tsetlin automaton with 2N states. The top row shows the state transitions that are made when the previous action resulted in a reward of 1; the bottom row shows transitions after a reward of 0. In states in the left half of the figure, action 0 is taken; in those on the right, action 1 is taken.

3.1.3. Strategy of Estimator

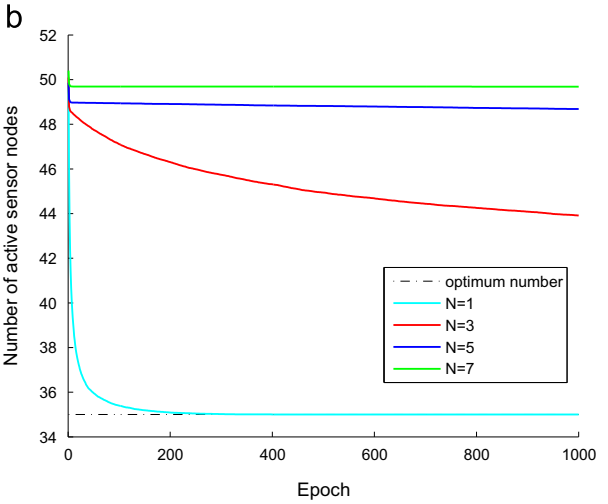
The strategy of updating Estimator is described by Algorithm 2, which works on the premise that the assumption of two properties given in Section 2 holds.

The interval $[lb, ub]$ represents the search region of possible k^*. Initially, the algorithm assigns the lower bound lb of possible interval of k^* as 0, and the upper bound ub as a sufficiently large integer (or ∞). Then it iteratively updates lb and ub depending on the feedback signal received from the base station, until lb equals to ub, which means that the Estimator has converged. The estimated value $\hat{k}^*$ equals to the one, which is farther from k_t, in lb and ub.

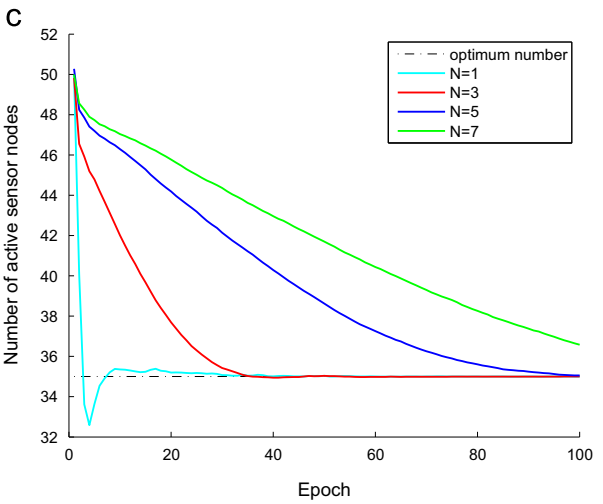
It is noted that the algorithm will keep running though it has converged. Once it is detected as a contradiction (a k_t less than $\hat{k}^*$ can satisfy the QoS requirement, or a k_t larger



Convergence time of original Goore Game with respect to different memory depth



Convergence time of Adaptive Goore Game with respect to different memory depth



Convergence time of Estimator Goore Game with respect to different memory depth

Fig. 4. (a) Convergence time of original Goore Game with respect to different memory depth. (b) Convergence time of Adaptive Goore Game with respect to different memory depth. (c) Convergence time of Estimator Goore Game with respect to different memory depth. Simulation results for different memory depth.

than or equal to $\hat{k}^*$ fails to satisfy the QoS requirement), the values of *lb* or *ub* will be reinitialized according to launch of a new self-adaptation. Therefore, the Estimator is able to track the optimum number of active sensors even when QoS requirement is varying with time.

3.2. Performance analysis

A theoretical analysis of the proposed algorithm's behavior is given in this section. We assume the memory depth $N=1$ in each algorithm, which means that the sensor will either stay in current state when rewarded or convert to the opposite state when punished. The reason for this assumption is twofold: (1) As Fig. 4 illustrates, all algorithms have superior performances with $N=1$; (2) Quick Convergence (QC^2) [16] is essentially a team of learning automata with memory depth 1.

In the following, we shall firstly prove that EGG is an absorbing Markov chain, with k^* being its absorbing state.

Theorem 1. *With a stable estimate value $\hat{k}^*$ and a diminishing parameter $\beta=1$, for any $\hat{k}^* \in \{0,1,2,\ldots M\}$, by taking the number of active sensors k as states, the EGG algorithm is an absorbing Markov chain, where $\hat{k}^*$ is the exclusive absorbing state and the others are transient states and M is the total number of the sensors deployed within the same cluster.*

Proof. The one step transition probability from state i to state j, $\forall i,j \in \{0,1,2,\ldots,M\}$ of EGG, is

$$P_{ij}=\begin{cases}\binom{i}{i-j}(1-r(i))^{i-j}r(i)^{j} & \text{for } i\geq\hat{k}^* \text{ and } i\geq j\\ \binom{m-i}{j-i}(1-r(i))^{i-j}r(i)^{j-i} & \text{for } i<\hat{k}^* \text{ and } i\leq j\\ 0 & \text{otherwise}\end{cases} \tag{2}$$

(a) We notice that $r(\hat{k}^*)=1$ and $r(k)>0.2$ for all k, then for any $j\in\{0,1,2,\ldots,M\}$, we have

$$P_{\hat{k}^*j}=\begin{cases}\left(\hat{k}^*\,\hat{k}^*-j\right)0^{\hat{k}^*}-j & \text{for } j\leq\hat{k}^*\\ 0 & \text{for } j>\hat{k}^*\end{cases} \tag{3}$$

Based on Eq. (3), it can be easily inferred that $P_{\hat{k}^*j}=1$ provided that $j=\hat{k}^*$, otherwise $P_{\hat{k}^*j}=0$.
Therefore $P_{\hat{k}^*\hat{k}^*}=1$, state $\hat{k}^*$ is an absorbing state of EGG.

(b) For any state i, we have

$$P_{i\hat{k}^*}=\begin{cases}\binom{i}{i-\hat{k}^*}(1-r(i))^{i-\hat{k}^*}r(i)^{\hat{k}^*} & \text{for } i\geq\hat{k}^*\\ \binom{m-i}{\hat{k}^*-i}(1-r(i))^{i-\hat{k}^*}r(i)^{\hat{k}^*}-i & \text{for } i<\hat{k}^*\end{cases} \tag{4}$$

From Eq. (4), evidently we always have $P_{i\hat{k}^*}>0$, which implies that from any state i it is possible to go to the absorbing state $\hat{k}^*$ within one step.
Based on the above and according to the definition of absorbing Markov chain [19], it is obtained that the EGG is an absorbing Markov chain. The proof is finished. □

Furthermore, we shall analyze the behavior of the proposed Estimator.

Theorem 2. *The estimate value $\hat{k}^*$ will converge to k* in any stationary environment. More exactly, given a time-invariant k*, with initial value $lb(0)=0$, $ub(0)=L$ (L is a sufficiently large number), we will have $\lim_{t\to\infty}lb(t)=\lim_{t\to\infty}ub(t)=k^*$.*

Proof. As described in Algorithm 2, although k^* is not known to us explicitly, we can update $\hat{k}^*$ by means of the feedback received from the upper-level application according to the following rules for two cases:

(1) Case $lb(t)\neq ub(t)$: If feedback is *unsatisfied*, then $lb(t)=\max(k_t+1, lb(t-1))$, we can derive

$$lb(t)\geq lb(t-1) \tag{5}$$

$$lb(t)=\max_t k_t+1 \quad \text{where } k_t<k^* \tag{6}$$

If feedback is *satisfied*, then $ub(t)=\min(\max(k_t, lb(t-1)), ub(t-1))$, we can derive

$$ub(t)\leq ub(t-1) \tag{7}$$

$$ub(t)=\min_t k_t \quad \text{where } k_t\geq k^* \tag{8}$$

In addition, we have

$$0=lb(0)\leq lb(t)\leq ub(t)\leq ub(0)=L \tag{9}$$

From Eqs. (5), (7) and (9), we can obtain that $lb(t)$ is monotonically increasing and bounded above, and $ub(t)$ is monotonically decreasing and bounded below.
So according to the Monotone convergence theorem, both $ub(t)$ and $lb(t)$ have a limit, we denote $\lim_{t\to\infty}lb(t)=\sup_t lb(t)$ and $\lim_{t\to\infty}ub(t)=\inf_t ub(t)$.
Furthermore, because $lb(t), ub(t), k_t\in\mathbb{N}$, from Eqs. (6) and (8), it is apparently that

$$lb(t)\leq k^* \tag{10}$$

$$ub(t)\geq k^* \tag{11}$$

As a result, we have $\sup_t lb(t)=\inf_t ub(t)=k^*$.

(2) Case $lb(t)=ub(t)$: As k^* does not vary with time, from Eqs. (10) and (11), we can obtain that $lb(t)=ub(t)=k^*$, which trivially proved the theorem.
We finished the proof by combining the two cases. □

Remark 1. According to Theorem 11.3 of [19]. In an absorbing Markov chain, the probability that the process will be absorbed is 1. Therefore, from Theorems 1 and 2, it is quite intuitive to infer that the EGG will eventually converge to the required QoS.

4. Experimental simulation

We run simulations to verify the effectiveness and efficiency of the proposed framework. In the simulations, 100 sensors are placed in an area of 100 m × 100 m randomly. We evaluate the performance of the proposed

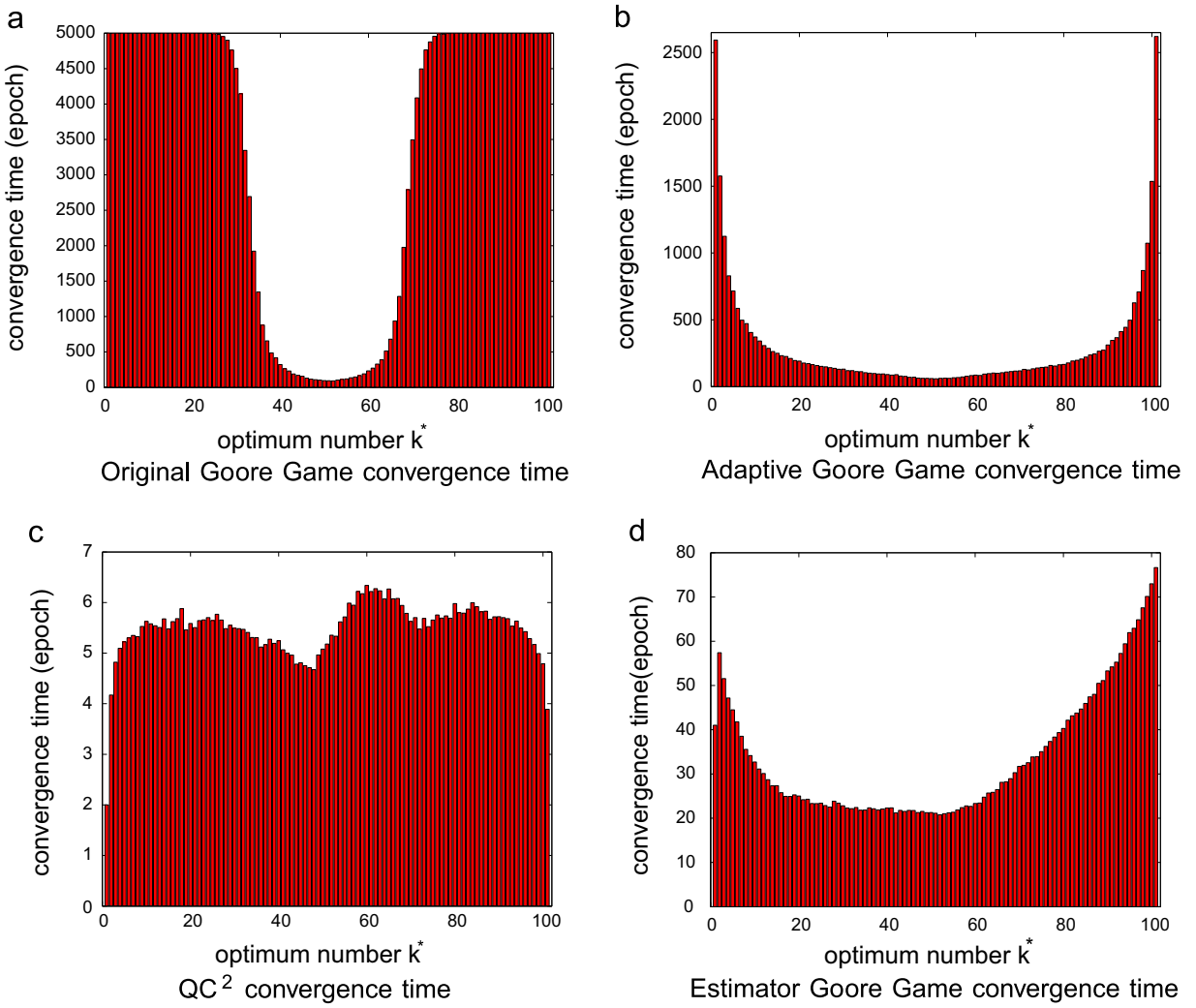


Fig. 5. (a) Original Goore Game convergence time, (b) adaptive Goore Game convergence time, (c) QC^2 convergence time, and (d) Estimator Goore Game convergence time. The convergence time distributed over the optimum sensor number ($\alpha = 0.95, \beta = 1$).

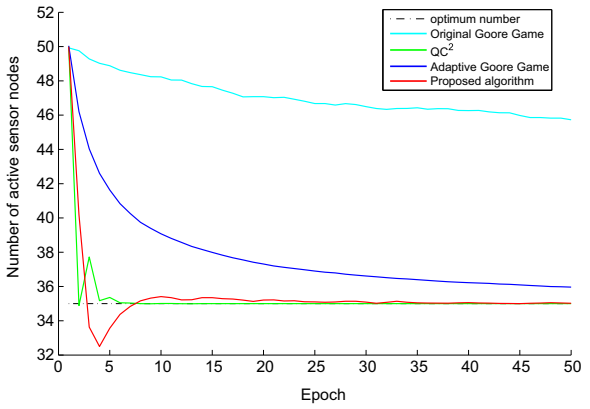


Fig. 6. The number of active sensors versus epochs with optimum number ($35\alpha = 0.95, \beta = 1$).

framework compared to those of three existing algorithm (original Goore Game, adaptive Goore Game and QC^2) from three different aspects — convergence time, network life and dynamic QoS tracking.

Table 1
Network life of OGG, AGG, QC^2 and EGG (results are averaged over 1000 independent simulations).

Algorithms	OGG	AGG	QC^2	EGG
Network life	7808	8637	1456	13,118
Improvement	NaN	10.6%	−81.3%	68%

4.1. Convergence time

Convergence time is defined as the first epoch t when k_t equals k^*. We compare the performance of original Goore Game, adaptive Goore Game, QC^2 and the proposed framework under different k^*, ranging from 0 to 100. All the results presented here are averaged over 1000 independent simulations. Figs. 5 and 6 show respectively the simulation results of the convergence time distributed over the optimum sensor number and the number of active sensors versus epochs with optimum number 35. From the figures, it can be seen that among all the four

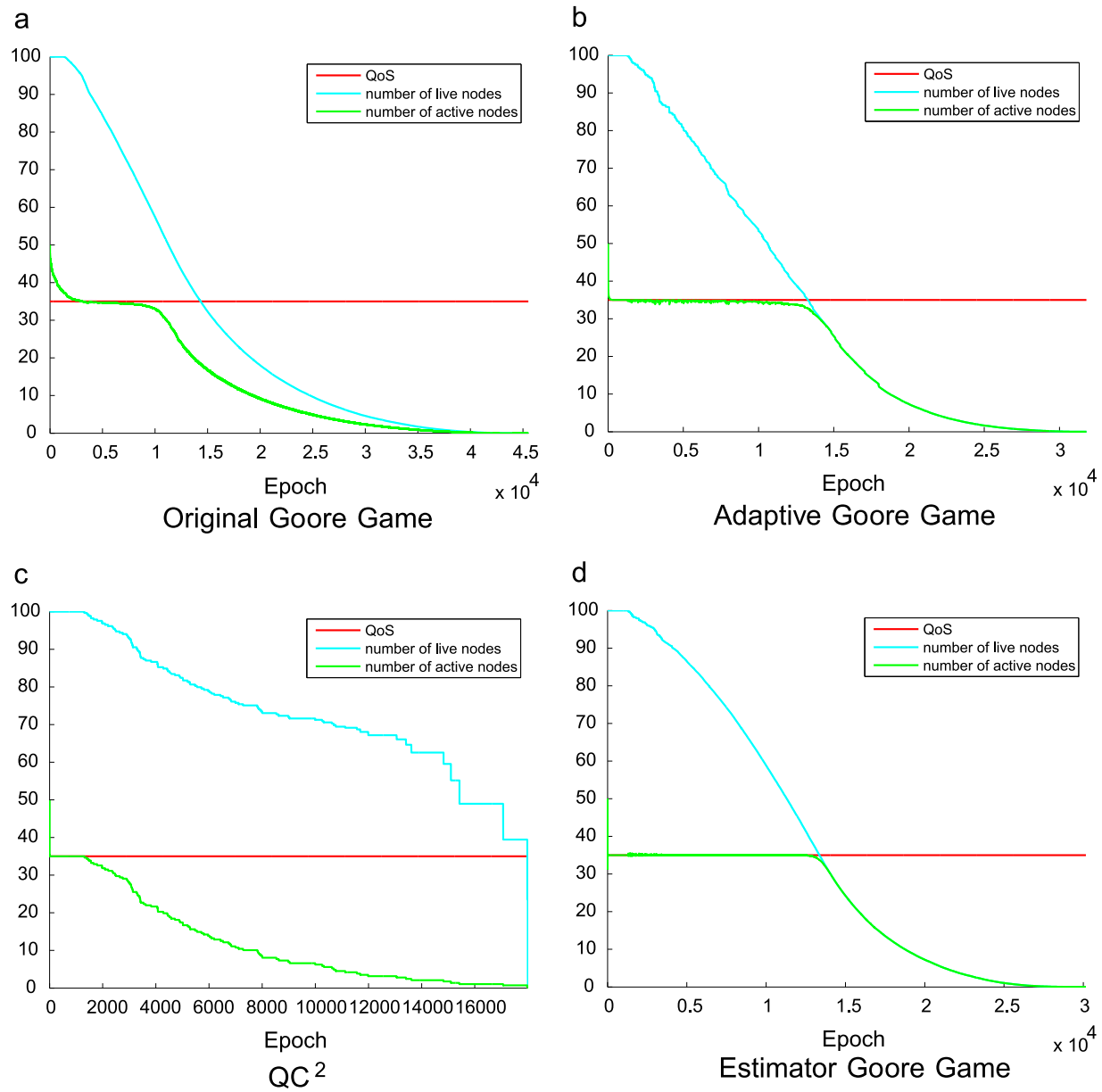


Fig. 7. (a) Original Goore Game, (b) adaptive Goore Game, (c) QC^2, and (d) Estimator Goore Game. The number of live and active nodes versus epochs ($\alpha = 0.95, \beta = 1$).

algorithms, QC^2 has the best performance of time convergence with different QoS requirement, and that its performance is almost not related to the selection of k^*. The proposed algorithm is the second best among all algorithms.

4.2. Network life

In this subsection, we shall evaluate the performance of the four algorithms with limited power supply. The terminology *network life* is defined as *the duration for which the desired QoS (i.e., the desired number of active sensors in the network) is maintained* in [13], and energy consumption model described in [20] was adopted for simulation. We assume that the base station (or sink node) has infinite energy, and each sensor node has 0.5 J energy in the beginning. The lengths of packets sent from a sensor node to the sink and from the sink to a sensor are 500 bits and 20 bits respectively. The optimum number k^* is set to be 35. The achieved network lives of the four algorithms are shown in Table 1. From Table 1, it can be seen that EGG is leading with improving the network life by 68%. The improvement is obtained with respect to network life of Original Goore Game.

Fig. 7 shows the result of the number of live and active nodes versus epochs, too. From the figure, we can conclude that EGG utilizes every live nodes if necessary. On the contrary, OGG and QC^2 do not make full use of all live

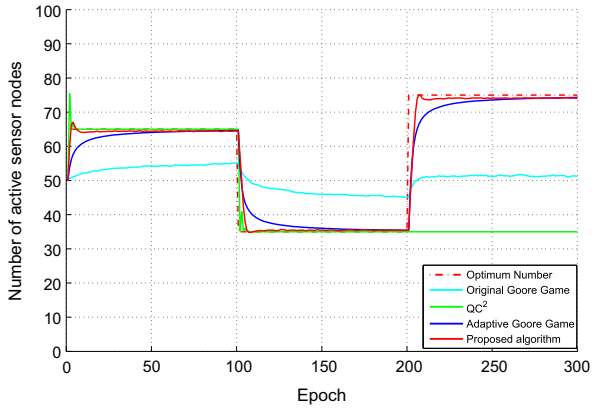


Fig. 8. The ability of tracking dynamic QoS requirement ($\alpha = 0.95, \beta = 0.99$).

nodes, consequently, they fail to reach the required QoS though the number of the live sensors is enough, which significantly shorten the network life.

4.3. Tracking dynamic QoS

The QoS requirement might vary with time in practical applications. If we are informed when the QoS requirement changes, we can simply restart the algorithm to let the number of active sensors converged to the new optimum number. However, the ability of tracking the optimum number of sensors is desired especially in some circumstances where the change of QoS requirement happens stealthily.

In this subsection, the four algorithms are compared in a fiercely changing environment. The requirement of QoS (optimum number of active sensors) is set to be 65 initially, and drops to 35 at epoch 100, finally increases to 85 at epoch 200. In fact, if we assume that the optimum number of active sensors is known to the base station (or sink node), then the base station (or sink node) will be aware of any tiny change of QoS requirement. Because in such scenarios, the change of QoS requirement is reflected by a change of optimum number of active sensors.

In order to make a fair comparison, restarting is not allowed in this simulation. The diminishing parameter α of our proposed algorithm is set to be 0.95 and perturbation parameter β is 0.99 in this simulation and all results are averaged over 1000 independent simulations. The number of active sensors versus epochs of different algorithms is illustrated to evaluate their ability of tracking dynamic QoS requirement, shown as in Fig. 8. It is obvious, from the figure, that OGG is unable to track the dynamic QoS requirement due to its low rate of convergence. AGG and EGG can converge to the optimum number asymptotically. QC^2 tracks the reduction of QoS requirement well, but it is unable to track the increment of QoS requirement.

5. Conclusion

In this paper, a novel QoS control framework for WSNs is proposed without the need for any prior information about optimum active sensor number, and an algorithm named Estimator Goore Game as its implementation is proposed. The proposed algorithm has a relatively low computational complexity and memory cost, and it can solve QoS control problem effectively without imposing an additional burden to the sensor nodes. Our simulation results have shown that the proposed approach is competitive on convergence time and dynamic QoS tracking, considering less information required, and outperforms significantly on network life maintenance.

The problem of QoS control for wireless sensor network with incomplete information remains largely opened. Many issues are worthy to be investigated, for example how to design a reward probability generating function that can adjust the active sensor number more efficiently; how to deal with the situation where the information distributed on each sensor is nonuniform; what if the upper level feedback could be probabilistically misleading, and so on. These issues will be addressed in our future work.

Acknowledgments

This research work is funded by the National Science Foundation of China (61271316), the National Social Science Foundation of China (14ZDB167), Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, and Chinese National Engineering Laboratory for Information Content Analysis Technology.

References

[1] C. Otto, A. Milenkovic, C. Sanders, E. Jovanov, System architecture of a wireless body area sensor network for ubiquitous health monitoring, J. Mob. Multimed. 1 (4) (2006) 307–326.
[2] A. Milenković, C. Otto, E. Jovanov, Wireless sensor networks for personal health monitoring: issues and an implementation, Comput. Commun. 29 (13) (2006) 2521–2533.
[3] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, N. Gordon, T. Bessell, M. Rutten, S. Jha, Wireless sensor networks for battlefield surveillance, in: Proceedings of the Land Warfare Conference, 2006, pp. 1–8.
[4] L. Yuan, G. Qu, Design space exploration for energy-efficient secure sensor network, in: Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors, 2002, pp. 88–97.
[5] S. Chouhan, R. Bose, M. Balakrishnan, A framework for energy-consumption-based design space exploration for wireless sensor nodes, IEEE Trans. Comput.-Aid. Des. Integr. Circuits Syst. 7 (28) (2009) 1017–1024.
[6] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, D. Culler, Design of a wireless sensor network platform for detecting rare, random, and ephemeral events, in: Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, 2005, p. 70.
[7] S. Gupta, A review of qos in wireless sensor networks, Int. J. Comput. Trends Technol. 21 (1) (2015).
[8] D. Chen, P.K. Varshney, Qos support in wireless sensor networks: a survey, in: Proceedings of International Conference on Wireless Networks, vol. 233, 2004, pp. 1–7.
[9] S. Kaur, R. N. Mir, Quality of service in wsn—a review, Int. J. Comput. Appl. 113 (18) (2015).
[10] A. Ganz, Z. Ganz, K. Wongthavarawat, Multimedia Wireless Networks: Technologies, Standards and QoS, Pearson Education, Upper Saddle River, New Jersey, 2003.
[11] F. Xia, Qos challenges and opportunities in wireless sensor/actuator networks, Sensors 8 (2) (2008) 1099–1110.

[12] R. Iyer, L. Kleinrock, Qos control for sensor networks, in: Proceedings of IEEE International Conference on Communications (ICC'03), vol. 1, 2003, pp. 517–521.
[13] J. Frolik, Qos control for random access wireless sensor networks, in: Proceedings of Wireless Communications and Networking Conference, vol. 3, 2004, pp. 1522–1527.
[14] L. Zhao, C. Xu, Y. Xu, X. Li, Energy-aware qos control for wireless sensor network, in: Proceedings of the First IEEE Conference on Industrial Electronics and Applications, 2006, pp. 1–6.
[15] M. Ayers, Y. Liang, Gureen game: an energy-efficient qos control scheme for wireless sensor networks, in: Proceedings of International Green Computing Conference and Workshops (IGCC), 2011, pp. 1–8.
[16] H.-L. Wang, W.-L. Hung, Qc 2: a qos control scheme with quick convergence in wireless sensor networks, ISRN Sens. Netw. (2013) http://dx.doi.org/10.1155/2013/185719.
[17] E.M. Elshahed, R.A. Ramadan, S.M. Al-tabbakh, H. El-zahed, Modified gur game for wsns qos control, Proc. Comput. Sc. 32 (2014) 1168–1173.
[18] K.S. Narendra, M.A. Thathachar, Learning automata: an introduction, Courier Dover Publications, Mineola, New York, 2012.
[19] C.M. Grinstead, J.L. Snell, Introduction to probability, American Mathematical Society, Providence, Rhode Island, 1998.
[20] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000, 10 pp.