Energy-Aware QoS Control for Wireless Sensor Network

Lei Zhao\(^1\)\(^2\), Chaonong Xu\(^1\)\(^2\)\(^3\), Yongjun Xu\(^1\)\(^2\), Xiaowei Li\(^1\)

1) (Advanced Test Technology Lab, Institute of Computing Technology, Chinese Academy of Sciences, 100080)
2) (Graduate University of Chinese Academy of Sciences, 100080)
3) (Department of computer science, Hefei University of Technology, Hefei PR China, 230009)

\{zl, xu_chaonong, xyj, lxw\}@ict.ac.cn

Abstract

While a lot of research has been done on some important aspects of WSN such as architecture and protocol design, supporting Quality of Service in WSN is still a largely unexplored research field. An application-specific QoS has been defined as the sensor network resolution. The mathematical Gur Game algorithm is used to achieve optimal active sensor number. In this paper, we design a novel energy-aware algorithm based on the previous work to solve the QoS problems. The periodical sleeping mechanism is introduced into the algorithm to save energy. When the optimal number sensors are achieved in initialization stage, not all of the remaining sleep-state nodes need to wake up every second. Simulation results show our method can save more energy than the previous.

1 Introduction

Recently there has been a lot of interest in building and deploying wireless sensor networks. Wireless sensor networks are autonomous ad hoc networks designed for some potential applications in environmental monitoring, surveillance, military, health, and so on. A distributed sensor network consists of many small, extensible nodes. Once the nodes are deployed throughout an area of interest, they collect data from the environment and automatically establish ad hoc networks to transmit their data to a base station (or sink node)\(^\dagger\)\(^\ddagger\). The base station aggregates and analyzes the report messages received and decides whether there is an unusual or concerned event occurrence in the deployed area. Energy source provided for sensor nodes is usually battery power. The sensor nodes must have a lifetime on the order of months to years, since it is undesirable or impossible to recharge or replace. Therefore, energy efficiency becomes the crucial design challenge in sensor networks.

Traditional QoS of network and application level (delay, throughput etc) for wireless communication systems and networks are not appropriate and adequate for wireless sensor network. In [3], the authors give a definition of QoS to spatial resolution (the optimum number of sensors that should be sending information at any given time). Based on the assumption that the base station communicates QoS information with sensor nodes using broadcast channel, the mathematical paradigm of the Gur game are used to dynamically adjust the optimum number of active sensors. For every second, all of the sensor nodes should be active to receive the reward probability from the sensor node. And then the rewarded nodes will keep active, others go sleep. So many sensor nodes should transit between two states every second. But sleep-state transition has the overhead (storing processing state and turning off power) and waking up also waste energy.

In this paper, we propose a new method to save energy which doesn’t need all sleep nodes waking up every second. Compared with the former work, we could bring down the whole energy dissipation without affecting the QoS. The remainder of the paper is organized as follows: Section 2 describes the Gur Game baseline algorithm used in sensor network and the relevant work. The improved algorithm of us is discussed in Section 3. The simulation result for the both algorithm are analyzed and compared in Section 4, and Section 5 concludes the paper.

2 Previous Works

2.1 Current Research on QoS for Wireless Sensor Network

There exists many envisioned applications in WSN and their QoS requirements may be very different. It is unlikely that there will be a “one-size-fits-all” QoS support for most applications. The present research efforts related to the QoS in WSN fall into three categories: traditional end-to-end QoS, reliability assurance, and application-specific QoS\(^\ddagger\). Sequential Assignment Routing (SAR) is the first protocol for WSN that includes a notion of QoS. The objective of the SAR algorithm is to minimize the average weighted QoS metric throughout the lifetime of the network\(^\ddagger\). Recently, K.Akayya proposed an energy-aware QoS routing protocol for WSN. The protocol finds a least cost and energy efficient path that meets certain end-to-end delay requirement during the connection. Both of the above solutions are based on the concept of end-to-end applications\(^\ddagger\).

Some end-to-end reliability issues in WSN are proposed in [7][8]. The novelty of their work is that they consider the need for information-awareness and adaptability to channel errors along with differentiated

---

This paper was supported in part by National Basic Research Program of China under Grant No.90207002, and in part by National Natural Science Foundation of China under Grant No.90207002.
allocation of network resources based on the criticality of data. The concept of their reliability is still based on end-to-end service.

In [3], QoS has been defined as the optimum number of sensors that should be sending information at any given time. The author exploited the mathematical paradigm of the Gur Game to dynamically adjust the optimum number of sensors. Another QoS definition is described as balancing the application reliability with the goal of energy efficiency [9]. The authors provide the application QoS through the joint optimization of sensor scheduling and data routing.

Next section, we will introduce the Gur Game algorithm and the previous QoS control based on the algorithm.

### 2.2 The Gur Game

The algorithm uses the mathematical paradigm of the Gur Game [10]. The basic idea of the Gur Game is described in the following paragraphs.

#### Fig. 1 Typical Gur Reward Function

Let us introduce the Gur Game with a simple example. Imagine that we have many players, none of whom are aware of the others, and a referee. Every second, the referee asks each player to vote yes or no, and then counts up the yes and no answers. A reward probability \( r = r(k) \) is generated as a function of the number \( k \) of players who voted yes. We assume that \( 0 \leq r(k) \leq 1 \). A typical function is shown in Fig. 1. Each player, regardless of how he or she voted, is then independently rewarded (with probability \( r \)) or penalized (with probability \( 1-r \)). A reward pushes the sensor to a higher state, i.e. toward +N or –N. The nodes at edge state self-loop. A punishment moves the sensor’s state towards the centre. When the desired QoS number achieved, the sensor nodes states will be stable. The sensor nodes in network has high die-off rate due to the limited battery power and bad circumstance. So if some

#### Fig. 2 Gur Memory of Size N=2

In [11], it is assumed that each player could have a memory of his previous trials. So a finite discrete-time automaton \( M_i \) is associated with each player \( i \). The finite state automation represents the player’s memory. It is a single (nearest-neighbor) chain of consecutive states where the total size of the memory is 2N, for some arbitrary N (which we refer to as the size of the automation). Starting with the leftmost state, we number the states from –N to 1, then from 1 to N (see Fig. 2). Note that this partitions the chain into a left half (with negative numbered states) and a right half (with positive numbered states). The player is allowed to be in only one state at any given time. Transitions exist only between nearest neighbor states \( j \) and \( j+1 \) and \( j-1 \)(i.e., the player can transition only to adjacent states).

### 2.3 QoS Control Based on Gur Game

QoS for wireless sensor network defined in [3] is spatial resolution (optimal number of active senor nodes that sending information to base station). The base station can communicate the QoS with sensor nodes through broadcast channel, and the mathematical paradigm of the Gur Game described above is used to dynamically adjust the optimum number of sensors. The sensor nodes act as game players, base station is a referee. Each sensor node can operate in one of two states: TRANSMIT or RECEIVE. When the nodes in TRANSMIT state, they send sensing data to base station and receive control feedback from it. In Receive state, the nodes only receive feedback from the base. The energy are saved for the RECEIVE state dissipate less energy than TRANSCIV. Associated with each sensor is a finite, discrete-time 2N-state automation which has N TRANSCIV states (+1…+N) and N RECEIVE states (-1…-N) as shown in Fig. 2 for the N=2. The base station has a reward function \( r(k) \). At each time, the base station count the number of packets \( k \) it has received to calculate the probability \( r(k) \). Then the base broadcasts this probability to all the sensors. Each sensor, in turn independently reward itself with the probability \( r(k) \) and punishes itself with probability \( 1-r(k) \). A reward pushes the sensor to a higher state, i.e. toward +N or –N. The nodes at edge state self-loop. A punishment moves the sensor’s state towards the centre. When the desired QoS number achieved, the sensor nodes states will be stable. The sensor nodes in network has high die-off rate due to the limited battery power and bad circumstance. So if some
nodes in TRANSMIT state die, the optimum QoS will decline accordingly, a new round trailing will begin automatically among all the nodes to get the optimum number. Another case is the base station changes the QoS requirement at some time, then the reward function must change the value accordingly. All nodes despite their state will start trial based on the algorithm above until the optimal number is achieved.

The two important goals of the author wants to accomplish are: 1) Have enough sensors in TRANSMIT state and sending packets toward the information sinks so that enough data is being collected (i.e. QoS); 2) maximize the lifetime of the sensor network by having sensors periodically power-down to conserve their battery energy. The network life is then defined as the duration for which the QoS is maintained. But we find that the author assume in a power-down state, a sensor can still receive and react to low-level signals. That means the power-down state is actually a RECEIVE state. It has been proved that the energy of sensor nodes dissipated in RECEIVE state is no less than the TRANSMIT state[2]. So the method periodically power-down couldn’t conserve battery energy. If we consider the power-down state as SLEEP, it will cost more less energy than RECEIVE. But there is another question that the nodes in SLEEP should wake up every second to receive the message from the base station. The transition energy consumption can’t be ignored as shown in FIG 3[12].

Fig.3 State Transition Latency and Power

3 Energy-aware QoS Control Algorithm for Wireless Sensor Network

In this section, we propose our algorithm in detail. We define 4 sensor node states in our algorithm: TRANSCIVE1 (T1), TRANSCIVE2 (T2), RECEIVE (R) and SLEEP (S). The components used in 4 different states are shown in Table 1.

<table>
<thead>
<tr>
<th>State</th>
<th>MCU</th>
<th>Sensor</th>
<th>Radio</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>On</td>
<td>On</td>
<td>Tx, Rx</td>
</tr>
<tr>
<td>T2</td>
<td>On</td>
<td>Off</td>
<td>Tx, Rx</td>
</tr>
<tr>
<td>R</td>
<td>On</td>
<td>Off</td>
<td>Rx</td>
</tr>
<tr>
<td>S</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
</tbody>
</table>

Table.1 Sensor Nodes States

The nodes in T1 state will send sensing information to the base station every second. In T2 state, the nodes only transmit control packets which a few bytes long compared with the thousands of sensing packets length without using sensing unit. R state means that the nodes only receive acknowledge packets from the base. In S state the node will power off all components and wake up at the scheduling time. When the sensor nodes send sensing or control information to the base station, there is a state flag in the packet to indicate the node’s state. The sensing and control information packet format is shown in FIG. 4.

Sensing Packet Format

<table>
<thead>
<tr>
<th>ID</th>
<th>Flag</th>
<th>Sensing data</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bit</td>
<td>2 bit</td>
<td>2000 bit</td>
</tr>
</tbody>
</table>

Control Packet Format

<table>
<thead>
<tr>
<th>ID</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bit</td>
<td>2 bit</td>
</tr>
</tbody>
</table>

Fig.4 Data Packet Format

Be different with previous work, two reward function are used in our paper: \( r_1(f) \) and \( r_2(f) \). \( r_1(f) \) is used for adjusting the nodes state between T1 and T2 to achieve the optimal QoS. We use \( r_2(f) \) to choose some nodes in T1 to sleep for energy saving. The initial value of \( r_1(f) \) in base station is equal to 0, and \( r_2(f) \) is 1. The base station will ignore the control information from T2 state nodes and keep \( r_2(f) \) be stable until \( r_1(f) \) is equal to 1. The format of acknowledge packet broadcast by the base station is shown at Fig.5:

Fig.5 Acknowledge Packet Format

The whole process of our algorithm can be divided into 2 stages: Initialization and Run stage.

A. Initialization stage

In Initialization stage, we should get the desired QoS firstly, and then choose the sleep nodes from the remaining. The work can be followed by three steps:

Step 1: For initialization, all nodes are in T1 state, and transmit sensing information to the base station. Assuming that the base desired QoS is 35, it means 35 nodes should be in T1 state. The Gur Game distributed control algorithm described above is used to adjust the nodes state between T1 and T2 until the preferred QoS are achieved. The probability broadcast back from the base is \( r_1(f) \):

\[
r_1(f) = 0.2 + 0.8 \exp(-0.002*(f - 35)^2)
\] (1)
At the time the probability $r_1(f)$ equals to 1, the desired QoS is achieved, and the state of all nodes will be in stable. Nodes in T1 will send sensing information every second, other nodes in T2 will have a new trial in step2. FIG. 6 shows the state transition between T1 and T2.

**Fig.6 T1 、 T2 State Transition (N=1)**

**Step 2:** In this step, new trial based on Gur Game algorithm will start among the nodes in T2 state. We choose 50 percent (it was noted in simulations that obtaining a rate of 50 percent was relatively easy—undoubtedly because when the voting is skewed one way or the other, even a small possibility of punishment rebalances the votes by shifting votes from the majority to the minority) of the T1 nodes to be the R state. When the base find the $r_1(f)$ is equal to 1, it will count the number of the control information it received and calculate the $r_2(f)$. The reward function $r_2(f)$ is:

$$r_2(f) = 0.2 + 0.8 \exp(-20*(f - 0.5)^2)$$  \hspace{1cm} (2)

Then $r_2(f)$ and $r_1(f)$ will be broadcast back to all nodes in one packet. Nodes in T2 only care the value of $r_1$, and $r_2$ will be accepted by nodes in T1. During the trial, the T1 state nodes will send the sensing information to the base every second as usual. And $r_1$ function value will keep a constant 1 in stage 2 no matter how many T1 nodes die off. When the $r_2$ is equal to 1 again, the algorithm will go into step3. The state transition diagram is shown in Fig.7.

**Fig.7 T2 、 R State Transition (N=1)**

**Step 3:** When the $r_2$ received is equal to 1, all nodes in R state will be scheduled to sleep for a constant interval (power off the sensing, communication, computing unit). And the nodes in T1 state will stop transmit control message.

**B. Running Stage**

There are 3 kinds of state the node use during the running stage, T1, T2 and S. Just as [3] described, the base station will investigate the QoS of the whole network in real time. If there are some nodes die in this stage due to the battery power depletion or other circumstance reasons, Gur Game algorithm will adapt automatically the nodes state between T1 and T2 instantly.

After the sleeping time expired, all nodes in S state will wake up to be the T1 state. The algorithm will go to initialization stage again. If the $r_1$ they received is 1, then the algorithm will turn to step2. Otherwise they will start from step1. When the $r_1$ and $r_2$ are equal to 1, the process goes to running stage again. If the base station want to get a new QoS, the reward function $r_1(f)$ must be adjusted. We propose that this work could only be done in initialization stage when the scheduled sleeping time expires.

### 4 Simulation and Analysis

Now we present the simulation of our algorithm. In the following, we assume that there are one master base station and 100 sensor nodes with no sensors failures or renewals. All nodes can directly communicate with the base station. Thus, we assume that the number of sensors does not change over time, and packets sent from the sensors arrive instantaneously at the base station. We also assume that feedback from the base station arrives instantaneously to all the sensors. The base station desires a rate of 35 packets received at each second, and let 50 percent of T1 state nodes to sleep. The two Gur reward function we used are $r_1(f)$ and $r_2(f)$, the trace of two function are shown as Fig.8 and Fig.9.

**Fig.8 Reward Function $r_2(f)$**

**Fig.9 Reward Function $r_1(f)$**

Then we run 2000 seconds for the control case of our algorithm based on the above parameter. Fig.10
show a trace of the number of packets received versus time. As one can see, the number of packets received by the base station fluctuates in the beginning but quickly converges to the optimal. Once there, it locks on as each sensor is rewarded with probability 1 and feedback is instantaneous. The time used of our algorithm to get the optimal is same to the baseline algorithm. The trace of the number of nodes in T2 state versus time is shown in Fig.11. When the optimal percent is achieved, the nodes in T2 state will go to sleep and wake up every second; the other will sleep a relatively long time and wake up every 500 second.

![Fig.10 Packets Received Vs Time](image1)

![Fig.11 Optimized Nodes Number Vs Time](image2)

We also compared the energy consumption of 3 different methods in our simulation. Table 2 shows the energy dissipation and states transition latency of 4 different states. If sensor nodes go to sleep and wake up every second to see whether the QoS are changed, the transition energy dissipation can not be ignored. As the Fig. 3 show, the energy dissipation including transition is given by[14][15]:

\[
E_t = P_k (t_i - \tau_{dk}) + \frac{P_o + P_k}{2} (\tau_{dk} + \tau_{ad})
\]

(3)

We also assume the radio dissipates \(E_{elec}=50\text{nJ/bit}\) to run the transmitter or receiver circuitry. Thus, to transmit or receive a k-bit message can be calculated as:

\[
E_{Tx}(k) = E_{Rx}(k) = E_{elec} \times k
\]

(4)

The sensing unit packets are 2010 bit long and 10 bit for control message. The packets received from the base are 16 bits long.

Total energy consumption can be calculated as:

\[
E_{total} = E_t + E_{Tx}
\]

(5)

<table>
<thead>
<tr>
<th>State</th>
<th>(P_k(\text{mw}))</th>
<th>(T_k(\text{ms}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>540</td>
<td>Not applicable</td>
</tr>
<tr>
<td>T2</td>
<td>350</td>
<td>20</td>
</tr>
<tr>
<td>R</td>
<td>300</td>
<td>25</td>
</tr>
<tr>
<td>S</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 2 Power Consumption and Latency

Table 2 list the power consumption of the sensor nodes in different states and the latency time of the state transition. Then the simulation was run for 5000 seconds for 3 different ways (Nodes in R state only; Nodes go sleep and wake up every second; half of the nodes wake up every second, the other every 500 second). Energy consumption is calculated on all nodes every second. The total energy dissipation of the whole network of 3 different methods is compared in Fig.12. \(O(t)\) indicates the energy consumption of our algorithm, \(S(t)\) and \(R(t)\) indicate the baseline algorithm when the remaining nodes in Receive or Sleep state. From the figure we can see that, our algorithm can save more energy than other two methods.

![Fig.12 Energy Consumption](image3)

5 Conclusion and Future Works

While a lot of research has been done on some important aspects of WSN, supporting Quality of Service (QoS) in WSN is still a largely unexplored research field. The algorithm using the Gur Game paradigm that allowed the base station to specify the optimal number of sensors has been proposed in [1]. In this paper, we introduce the periodical sleeping into the algorithm to save energy. When the optimal number sensors are achieved in initialization stage, not all of the remaining S state nodes need to wake up every second. We use the Gur Game algorithm to select half of the nodes sleep for a relatively long time, others wake up every second. Simulation result shows that our methods can save more energy than other two.

In this paper, we have assumed that each sensor can communicate directly with the base station, and a broadcast channel between the base and the sensors. It is not very scalable to real-life sensors. In our future works, we will consider integrate the low-energy adaptive clustering hierarchy with the base Gur algorithm to achieve scalability. Also, we consider...
introduce the energy model and fail probability model to the sensor node to find how many nodes in deep SLEEP state for how long.

References


