DUTY CYCLING IN TARGET MONITORING WIRELESS SENSOR NETWORK SYSTEMS

Ying Guo[1], Feng Hong[2*], Zhongwen Guo[2]

[1]Qingdao University of Science and Technology

[2]Ocean University of China

Qingdao, China

Emails: [1]guoying@qust.edu.cn, [2*]hongfeng@ouc.edu.cn, [2]guozhw@ouc.edu.cn

Submitted: Jan. 15, 2013 ***Accepted: May 21, 2013*** ***Published: June 2013***

Abstract- Duty cycling is an important method for energy constrained sensor network systems to prolong lifetime. Current research on duty cycling is mainly based on such assumption that all the sensory coverage should be maintained while some nodes are sleeping. However, for the applications of target monitoring, it is not necessary anymore to keep the whole sensory coverage of the sensor network. It only needs to make sure that such kinds of nodes are active which can perceive the activities of the monitored targets. This paper proposes a novel duty cycling design for target monitoring sensor networks, which includes two algorithms - sleep scheduling algorithm and gradient query algorithm based on sleep periods. In the proposed design, most of sensor nodes are sleeping, while still keep the functions of target monitoring and information query in the sensor networks. The performance of our

design has been evaluated through both theoretical analysis and simulations, which prove the functionality of the proposed design on the reduction of energy consumption.

Index terms*:* Duty cycling, sleep scheduling, gradient query, target monitoring, wireless sensor network systems.

I. INTRODUCTION

Wireless sensor network systems are deployed in the physical environments to monitor or detect targets, for which the constrained energy supply of the batteries of sensor nodes is one fundamental bottleneck. Prolonging the lifetime of sensor networks is a prime consideration in network design [1, 2].

Current approaches of duty cycling allow some nodes to go to sleep without destroying sensory coverage. In order to satisfy the appropriate degree of coverage, certain number of nodes must keep awake [3, 4, 5]. Both random and synchronized sleep scheduling algorithms have been studied in depth [6, 7]. However, one important factor has been left out of consideration in the approaches mentioned above, which is that it does not always need to keep the sensory coverage for some certain kinds of applications. For example, the applications of target monitoring do not need all the deployment area be covered by sensor nodes. They only need to make sure that such kinds of nodes are active which can perceive the activities of the monitored targets, while a large number of the sensor nodes can go to sleep [8].

In order to prolong the lifetime of this kind of sensor networks as much as possible, two problems have to be solved in the design of duty cycling mechanism: (1) As the targets are moving in the deployment area, how to keep the targets being sensed while most of the nodes are in sleep. The key issue contains two aspects, which can be divided as what kinds of sensor nodes can go to sleep and how long they can sleep. (2) In the target monitoring sensor networks, another nontrivial problem is how to transfer the target information to users when most of nodes are sleeping.

In this paper, we present a novel duty cycling design fort he sensor networks of target monitoring. Our design includes two algorithms, which are Sleep Scheduling Algorithm (SSA) and Gradient Query Algorithm (GQA) based on sleep period. SSA is exploited to find out the optimal sleep

periods for every node to save energy and to ensure that the targets always be monitored when they are moving, while most of the nodes are in sleep mode. GQA is to guarantee that all the nodes have to be wakening up when the user queries are answered through the gradient query route. The gradient of GQA is based on the sleep periods of sensor nodes, while the target information is always stored in active nodes. Meanwhile, the route of GQA is the shortest path from the user to the information source through sleep periods decreased path.

Major contributions of this paper can be summarized as follows:

- We propose a novel duty cycling design for the target monitoring sensor network systems, which can prolong the network lifetime as far as possible, and can respond the target information queries through gradient query routing.
- The performance of the proposed design has been evaluated through theoretical analysis and simulations, which prove functionality of the proposed design on the reduction of energy consumption.

The rest of this paper is organized as follows. Section II summarizes the related works on sleeping schedule and gradient query. Section II and IV present the design of SSA and GQA in details and analyze their performance theoretically. Section V discusses the case of multi-targets monitoring of the proposed design. Section VI presents simulations and demonstrates the performance of SSA and GQA. Section VII concludes this paper.

II. RELATED WORK

Lots of methods about duty cycling have been proposed in recent literatures. The energy consumption in sleep state is much smaller than in active and listen state, so taking nodes to sleep is an efficient method to save energy. Most of researches focus on taking nodes to sleep while maintaining full coverage, such as [3, 4, 5]. Some of them focus on solving the problem of set k-coverage [9, 10], which divide the network area into several subsets, and each of the subsets can cover the whole monitoring area. So these approaches only keep one of the subsets in active state to satisfy full coverage. [11, 12] make use of the positional relationship of neighbor nodes and let redundant nodes to sleep. These methods need more than enough nodes to be deployed to guarantee sensing coverage, as they are based on a specified degree of redundancy.

Some researches focus on partial coverage [6, 7, 13]. These schemes are investigated to improve energy-efficiency. In the research of [6], each node chooses sleep and wake up times independently and randomly. In [7] all nodes go to sleep and wake up in a synchronized fashion. Both of them focus on the problem of tracking moving targets, while [13] focuses on stationary target detection and proposes a localized algorithm that approaches the minimum average delay bound.

SSA is different from previous works, which makes use of the characteristic of target monitoring. All of the approaches mentioned above ignore the actual situation of target monitoring. They all focus on the coverage of the deployment area when some nodes are asleep. But in applications of target monitoring, it dose not need all the deployment area to be monitored, only the nodes which perceiving the target keep alive and others can go to sleep. As most of the nodes are in sleep state and no redundancy nodes to be deployed, SSA could save a lot of energy.

Many approaches about information storage and query in sensor network systems have been proposed [14, 15]. In directed diffusion, information discovery through a reactive approach [16], which allows the query node to flood its interests in the network to search the relevant data. As in TinyDB [17], the discovery of the desired information relies on flooding the network while little collaborative preprocessing is performed.

Flooding is wasteful compared with other schemes, thus alogical brokerage structure has been imposed to avoid flooding, which enables queries to rendezvous with data in the network. Geographical hash tables (GHTs) [18] maps the target type to a geographical location by a content based hash function. However, the data retrieval scheme in [18] is not distance-sensitive. Improvement of the flat hashing by hierarchical hashing has been investigated with hash locations aware of data correlation in [19, 20]. Similar data is stored closely, nearby users can discover source nodes more quickly. Double rulings [21] proposed a new scheme, which stores data replica at a curve instead of one or multiple isolated sensor sand the user travels along another curve which guarantees to intersect with the source curve.

Most gradient based routing [22, 23] uses the natural gradients of physical phenomena, which are based on the spatial distribution of many physical quantities, such as temperature, illumination and so on. Gradients imposed by natural laws can be far from perfect guides, as witnessed by the existence of local extreme or large plateau regions [24, 25, 26]. Because forcing information-guided routing to deteriorate to a random walk [27]. Information gradients in [27] diffuse in-

formation away from source nodes holding desired data to establish information potentials. Network queries can navigate along these information potentials and reach the sources. [28] develop a gradient based routing scheme such that from any query node, the query message follows the signal field gradient or derived quantities and successfully discovers all iso-contours of interest.

The novelty of GQA is that it dose not use natural gradient or processed information gradient, but makes use of sleep scheduling. GQA uses different sleep periods of nodes to make up sleep periods decrease gradient routes. Queries from users walk along these routes to get the data they interested. No redundancy transmission in the query process, because the route is the shortest path from the user to the source node.

III. Design of SSA

SSA is designed for distributed target monitoring sensor network systems. There are two phases to implement it, which are initialization phase and state transformation phase. In the following subsections, we introduce these two phases in details and optimize its sleep period.

a. Initialization

When nodes are deployed in the monitoring area, initialize time *T*i and time slot *T*u should be determined in advance.

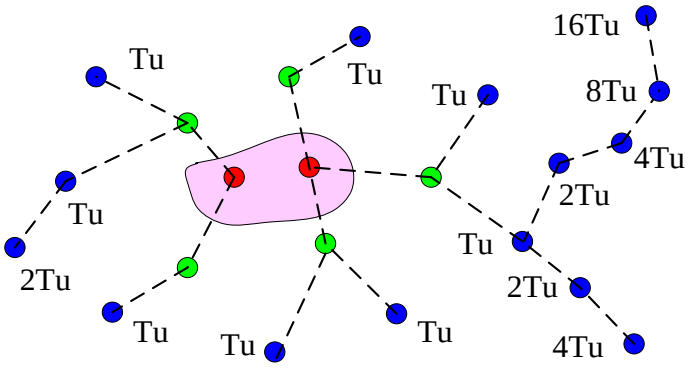


Figure 1. Initialization phase

There are three states for sensor nodes: active state, listen state and sleep state. Every node should decide its state in initialize time Ti. If it should turn to sleep state, it also has to compute out the length of sleep period. Nodes decide their states and sleep periods as follows:

After deployment, all the nodes search the target. If one node finds the target, it turns into active state and broadcasts the target (T) to others. The node which received the message (T) and doesn't detect the target, turns to listen state and broadcasts the target and the number of hops from it to the target (T, n) to others, where n=0 for listen state node. The node which received the packet (T, n) and doesn't detect the target turns to sleep state, and sets the sleep period as Tn. In order to make the nodes to sleep as long as possible and answer user queries in time, we set Tn $=2^n T$u. Then it makes the hop n plus one, and broadcasts the target and the new hop (T, n). This phase continues until reaching the bound of the monitoring area. When the time Ti passed, all the sleep state nodes turn to sleep with their own sleep periods.

The sleeping schedule does not only ensure the node farther to the target has the longer sleep period, but also ensure that when one node wakes up, all nodes nearer the target are waken, so it can update its state timely.

In order to guarantee all the nodes know their states, Ti must be long enough. Assuming the distance between the farthest two nodes of the deployment area is Dmax, the information propagation speed is Vi, so Ti=Dmax/Vi. These sleep state nodes turn to sleep at the time Ti after deployed. The time slot Tu should meet the following theorem:

Theorem 1: If the farthest two nodes of the deployment area has the largest number of hops Hmax and the distance between them is Dmax, the maximum moving speed of the target is Vmax, then the optimal time slot Tu is $\frac{D\max}{V\max 2^{(H\max-1)}}$.

Proof. The optimal time slot must match two important conditions, first is guaranteeing the monitor of the moving target, second is the sleep slot should as long as possible to save energy.

In order to guarantee the monitor of the moving target, the node can not wake up after the target pass by. Assuming the farthest two nodes has the largest number of hop Hmax and the distance between them is Dmax. Because these two nodes have the largest hops, if the target is detected by one of them, the other one will get the longest sleep period. In order to monitor the target, the sleep node must wake up before the target moving to it with maximum speed Vmax.

$$2^{(H\max-1)} \le \frac{D\max}{V\max} \quad (1)$$

*H*max is decreased by 1, because there is a hop from active state to listen state node, which is not included in sleep state. We can get *T*u from

$$Tu \le \frac{D\max}{V\max 2^{(H\max-1)}} \quad (2)$$

The large sleep period can reduce energy consumption. So time slot *T*u should take the largest value.

$$Tu = \frac{D\max}{V\max 2^{(H\max-1)}} \quad (3)$$

□

As shown in Fig. 1, the red and green nodes are in active and listen states respectively, and the blue nodes are in sleep state with different sleep periods.

In initialization phase, the red nodes detect the target and turn to active state. They broadcast the target (T) to their one hop neighbors. The green nodes receive the packet (T) and turn to listen state. They broadcast the target and hop (T, n) with n=0. The blue nodes receive (T, n) and compute out their sleep periods with n. They add one to n and broadcast (T, n) until all the nodes know their states. The blue nodes will go to sleep with their sleep periods after initialize time *T*i.

b. State Transformation

State transforms among active state, listen state and sleep state with the target's moving. The node at active state and sleep state can only transform to listen state, and the node at listen state can transform to both active state and sleep state, as in Fig. 2. Assuming the time used to calculate node's state is *T*c.

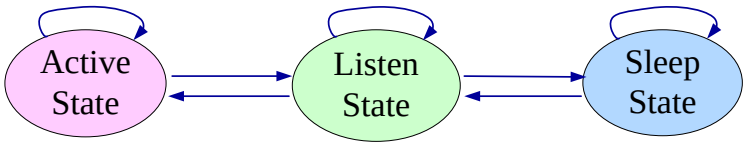


Figure 2. State transformation phase

Theorem 2: If the number of hops between a node and the target is n, then the sleep period *T*n of the node is 2^n*T*u-*T*c in state transformation phase.

Proof. In order to update the node state in time, the sleep period must ensure that when one node wakes up, all nodes nearer the target are waking. In the initialization phase, sleep period equal to $2^n Tu$ could meet requirements. But in the transformation phase, the node need some time to calculate its state, so the sleep period should reduce Tc, which is $Tn = 2^n Tu - Tc$.

□

The details of state transformation phase are as follows: The node at active state monitors the target, if it can not detect the target, it broadcasts the target has gone and turns to listen state. When the active state node receives other nodes' message of the target has gone, if it still can detect the target, it keeps in active state and broadcasts the packet it received to others, this information will be propagated until it meets the listen state nodes.

Algorithm1 (Nodes in active state)

```
01. Detecting the target;
02. if(Target=None)
03.     Broadcast(Target has gone);
04.     Send the target data to active state nodes;
05.     Goto listen state;
06. endif
07. if(Target!=None)&(Receive(Target has gone))
08.     Transmit received message;
09. endif
```

Usually, the node at listen state does not monitor the target. It only receives the information from active state nodes. When it receives the message about the target has gone away, it monitors the target to see if the target can be sensed. If it discovers the target, it turns to active state and broadcasts the target (T). Then, if it does not receive the response of sleep state nodes, it will broadcast the target (T) at next time slot again. If it does not discover the target but receives the packet (T) from its neighbors, it still keeps in listen state.

If the node does not discover the target and has no neighbors in active state, it will go to sleep state. It sets its sleep period as $Tn = Tu - Tc$ and broadcasts the target and hop (T, 1) to others. If it

receives the response of sleep state nodes, it will turn to sleep state at Tc+1, and if it does not receive the response of sleep state nodes, it will broadcast the packet again at next time slot. After getting the response, it may turn to sleep state at Tc+1 with sleep period Tn.

Algorithm2 (Nodes in listen state)

```
01. Receive(Target has gone);
02. if(Target!=None)
03.     Broadcast target (T);
04.     while(Receive=None)do
05.         Broadcast(T) at next time slot;
06.     Goto active state;
07. elseif(Neighbor!=active)
08.     Set sleep period;
09.     Broadcast(T, 1);
10.     while(Receive=None)do
11.         Broadcast(T,1) at next time slot;
12.     Goto sleep state at Tc +1;
13. endif
```

The node in sleep state wakes up when the sleep time is out. If it receives the target (T), it will respond the broadcasted node and turn to listen state. It broadcasts (T, 0) at every time slot until receives sleep state nodes' response. If it receives the target and hop (T, n), it responds the broadcast node and updates its sleep period with $T\text{n} = 2^{n}T\text{u} - T\text{c}$, then it broadcasts the target and its current hop (T, n) with $n=n+1$ to others. This sleep period could be used in the query process, but the node's real sleep period has to do some adjusting. If the new hop received smaller than the old one, the node goes to sleep at Tc+1 with this sleep period. If the new hop bigger than the old one, in order to guarantee the nodes at n+1 can update their states in time, the hop's change can not bigger than one. If it receives the response of nodes at n+1, it goes to sleep at Tc+1 with new Tn. If it does not receive the response, the node will go to sleep at Tc +1 with the old sleep period.

Algorithm3 (Nodes in sleep state)

```
01. if(Receive=(T))
02.     Broadcast(T, 1),until receive response;
03.     Goto listen state;
04. endif
05. if(Receive(T, n))
06.     Respond the broadcast node;
07.     Update sleep period with Tn;
08.     Broadcast(T, n);
09.     if(n<=the old one)
10.         Goto sleep with Tn at Tc +1;
11.     else
12.         if(Receive=None)
13.             Change n back to the old one;
14.             Goto sleep with old sleep period;
15.         else
16.             n=n+1;
17.             Goto sleep with Tn;
18.         endif
19.     endif
20. endif
```

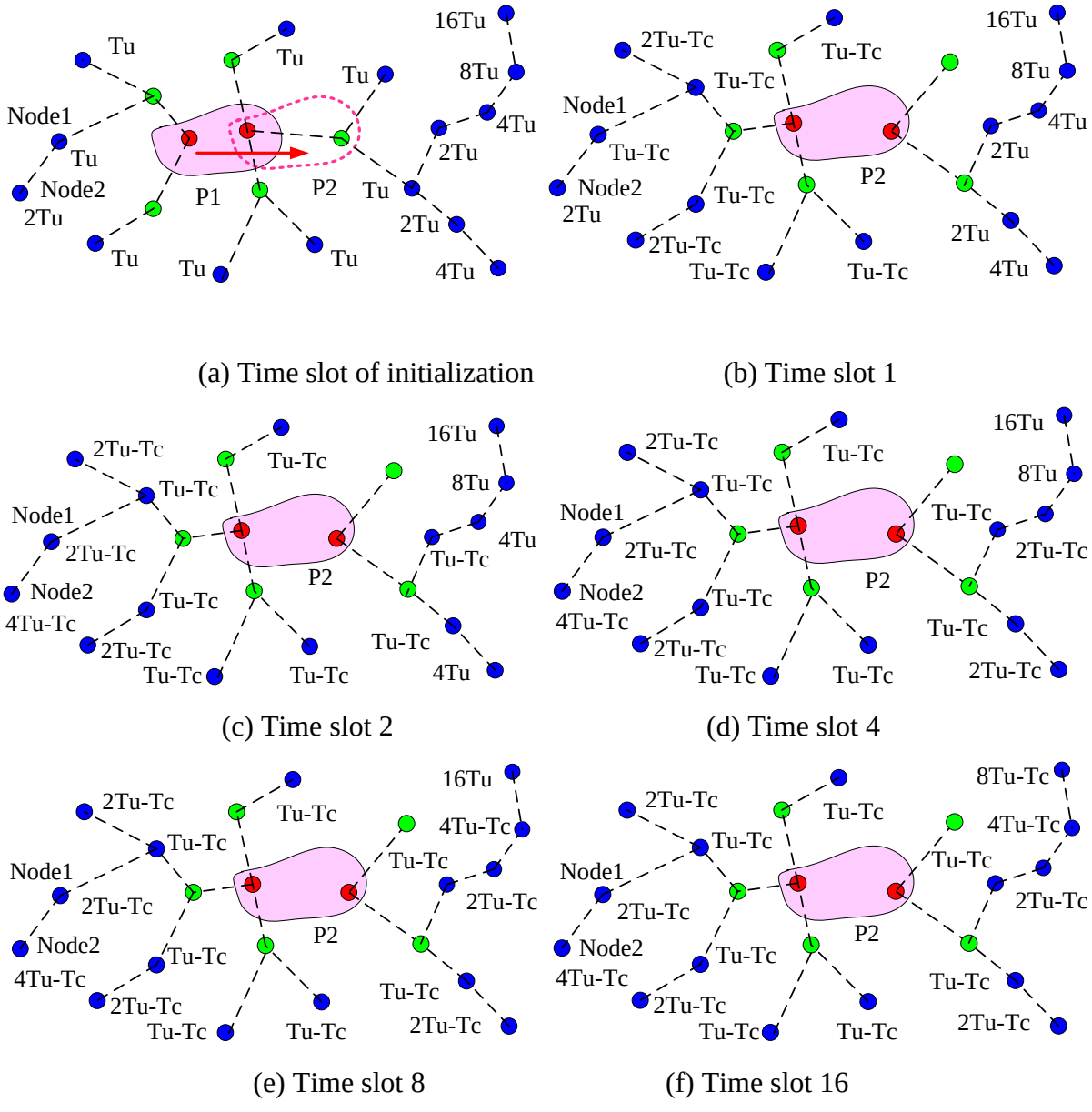
The priority of these three states is {active > listen > sleep}. One node may compute out more than one state with the packets it received, in this situation, it chooses the message with the highest priority as its state. If the node in sleep state wakes up and receives more than one packet with different number of hops n, it chooses the smallest one of them.

Then we analyze the time of SSA needed for all the nodes update their states theoretically.

Theorem 3: If the hop between the farthest two nodes has the largest value $H\max$, then the longest latency for all the nodes update their states is $2^{(H\max-1)}T\mathrm{u}$.

Proof. The longest latency for all the nodes to update their states is the time of the farthest node from the target updates its state. Assuming the farthest two nodes have the largest number of hops *H*max. And the target detected by one of them, another node will has the longest sleep period $2^{(H\max-1)}$*T*u, so this time is the longest latency for all the nodes to update their states.

□



(a) Time slot of initialization (b) Time slot 1

(c) Time slot 2 (d) Time slot 4

(e) Time slot 8 (f) Time slot 16

Figure 3. Initialization and state transformation phase

We can see the example of initialization and state transformation phase in Fig. 3. Red nodes are in active state, green nodes are in listen state and blue nodes are in sleep state with different sleep periods.

Fig. 3(a) is the initialization state, every node has computed out its state. The target moves along the red line from position 1 (labeled as P 1) to position 2 (labeled as P 2). In Fig. 3(b), the target has moved to position 2, all the nodes in active and listen states change their states in the first time slot with the target's position change. Node 1 (labeled as Node 1) receives n=1 and computes out its sleep period is Tn=2Tu−Tc, but at that time scale node 2 (labeled as Node 2) does not wake up, so node 1 can not receive the respond of node 2. So it gets to sleep with Tn=Tu-Tc at Tc+1. At the second time slot, the nodes in sleep state with sleep period equal to $2^i T$u (i=0,1) wake up and update their states, as in Fig. 3(c). At this time node 1 receives the respond of node 2 and gets to sleep with Tn=2Tu-Tc. And so on, other nodes update their sleep periods at time slots 4, 8 and 16 separately, as in Fig. 3(d), (e) and (f). At time slot 16, all the nodes finish the state transformation phase.

c. Optimize Sleep Period

The sleep periods mentioned above can guarantee the target monitoring, however, the nodes near the target wake up frequently which will cost superfluous energy. These nodes do not need wake up so frequently. Assuming the average distance between two nodes is Dave, which could be obtained from network topology after deployment. The hop number from a node to the node sensing the target is n hop, the longest sleep time of this node is

$$T\max = (n_{hop} - 1) \cdot \frac{Dave}{V\max} \qquad (4)$$

We use Tmax to optimize sleep periods of SSA.

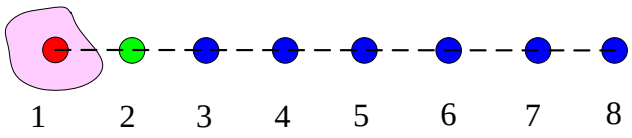


Figure 4. Example of optimization

When one node gets its hop number n to the node sensing the target, it uses n to compute out the sleep period Tn and the longest sleep time Tmax. It sets m=n+1=n_{hop}-1 because listen state node's

n=0. It compares Tn and Tmax, if Tn=Tmax, Tn is the sleep period. If Tn<Tmax, it sets m=m+i (i=1, 2, 3...) and uses m replace n to compute out new Tn, compares with Tmax until Tn≥Tmax. If Tn=Tmax, Tn is the sleep period, if Tn >Tmax it computes out sleep period Tn with m=m−1. The algorithm of sleep period optimization can be described as follow:

Algorithm4 (Sleep period optimization)

01. if(Tn<Tmax);
02. m=n+1,i=0;
03. while(Tn<Tmax)do
04. i=i+1;
05. Compute out Tn with m=m+i;
06. if(Tn>Tmax);
07. Compute out Tn with m=m-1;
08. endif
09. endif
10. Sleep period is Tn;

Take an example, as shown in Fig. 4, there are eight nodes deployed in line. The distance between every two nodes is 10m and the maximum speed of the target is 1m/s, and we set Tc=1s.The target at pink area sensed by node 1. So node 1 is in active state, node 2 is in listen state and all the other nodes are in sleep state.

Use Theorem 1, we can compute out Tu by $\frac{D\max}{V\max 2^{(H\max-1)}}$ =1s.We compare the sleep periods before and after optimization in Table 1.

Table 1: Sleep periods before and after optimization (s)

Node ID	1	2	3	4	5	6	7	8
Before Opt	Active	Listen	1	2	4	8	16	32
After Opt	Active	Listen	8	16	16	32	32	32

We assume the node in sleep state does not cost energy while in all other states costs one unit energy per second. We compare the energy cost before and after optimization within 70s, as shown in Fig. 5. Optimization saves 162/276=58.7% of energy.

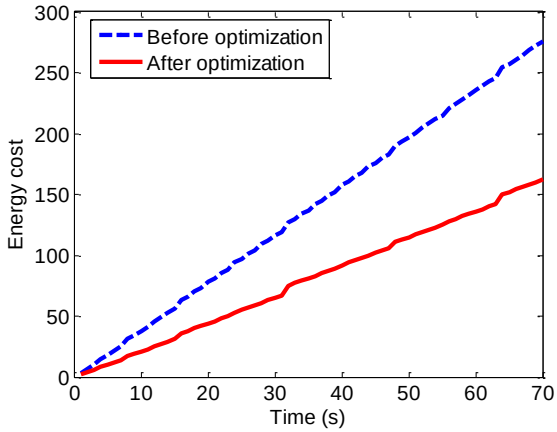


Figure 5. Energy cost in 70s

As we can see, the sleep period after optimization is much longer than before, and the nodes near the target do not wake up frequently. This will save more energy than before optimization, and could also ensure when one node wakes up all the nodes near the target also wake up, so it can update its state timely. The longest latency for all the nodes update their states is still as described in Theorem 3.

IV. DESIGN OF GQA

All of the target's information is stored in active state nodes. When an active state node wants to transform to listen state, it broadcasts its target information to other nodes in active state, as in Algorithm1 step 4. Therefore if one node wants to know the target information, it only has to find active state nodes.

The sleep scheduling guarantees that when one node wakes up, all the nodes nearer the target are waking. After updating the state information, sleep periods from the node who wants to know the target information to the active state nodes are decreasing. So the node sends the query packet along the sleep periods decreasing route to find active state nodes, at the same time it will find the information it needs.

Theorem 4: The sleep periods decreasing route has the shortest number of hops from the node who wants to get the information to the target.

Proof. When one node wakes up, it updates its state first. If the node is in sleep state, it gets its new number of hops to the target. The node chooses the shortest value from the hops it has received to compute out its sleep period. So the query goes to the target directly. The hops between them are the shortest number of hops. The query walks alone the sleep periods decreasing route, which means walking alone the hops decreasing route without detour. We can also say the sleep periods decreasing route is the shortest path from the node who wants to get the information of the target.

□

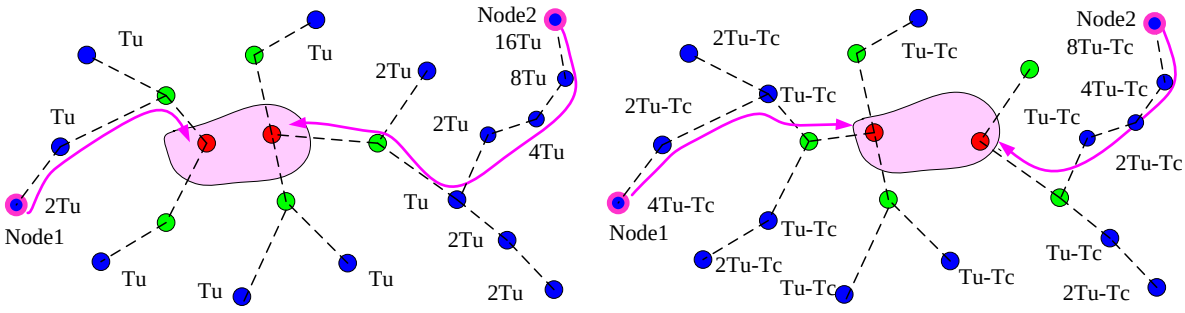
Then we analyze the latency of GQA for the nodes to get the target information theoretically.

Theorem 5: If the hop between the farthest two nodes has the largest value $H\max$, then the longest latency for the node to get the target information is $2^{(H\max-1)}T\mathrm{u}$.

Proof. The node can only get information when it wakes up, so the latency is equal to its sleep period. The longest sleep period is the longest latency. From Theorem 3 we know the longest sleep period is $2^{(H\max-1)}T\mathrm{u}$, which means the longest latency is also $2^{(H\max-1)}T\mathrm{u}$ in GQA.

□

The latency of GQA is equal to the sleep period. The maximum latency is $2^{(H\max-1)}T\mathrm{u}$. In the applications of target monitoring, the target information is not necessarily all the time. So this latency could be accepted in these applications.



(a) Time slot of initialization (b) Time slot 16

Figure 6. Sleep periods based gradient query phase

As we can see in Fig. 6, nodes in pink rings want to know the target information. The target information is always stored in active state nodes, which are labeled as red nodes. Query packets walk along the sleep periods decreasing routes, and reach active state nodes to get the target information, these routes are labeled in pink lines. Fig. 6(a) is the time slot of initialization, node1 can get the target information at every time slot 2*n*T*u* (*n*=1, 2, 3...), while node 2 can get the data at every time slot 16*n*T*u* (*n*=1, 2, 3...). In Fig. 6(b), node 1 and node 2 can get the information of the target at every time slot 4*n*T*u* and 8*n*T*u* (*n*=1, 2, 3...) respectively.

V. MULTI-TARGETS

Usually, there is more than one target to be monitored in the network. In this case, our method still works well.

In the initialization phase, every node only markets the first received notification and ignores others. In the transformation phase, if one node gets more than one state form different targets, the priority of these three states is {active>listen>sleep}, it chooses the highest priority state it has received. If the node is in sleep state, and receives more than one hop information from different targets, it chooses the smallest one.

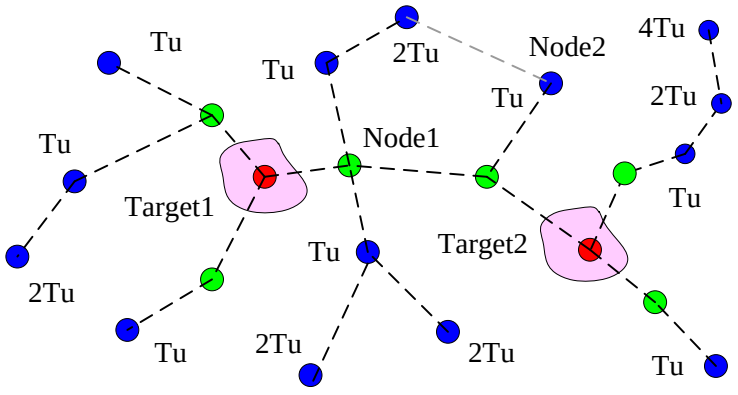


Figure 7. Node states of multi-targets

For example, there are two targets in Fig. 7, node 1 (labeled as Node 1) gets the listen state from target 1 (labeled as Target 1) and hop=2 from target 2 (labeled as Target 2), it chooses listen state of target 1, because listen state's priority is higher than sleep state. Node 2 (labeled as Node 2) receives hop=4 from target 1 and hop=2 from target 2, as 2<4, it computes out its sleep period with hop=2 of target 2.

The target information is stored in active state nodes of the target it be longs to. And the query walks along the sleep periods decreasing route to find the target. But it may find the target it does not interesting, in this case, it has to find out another route to get the information it needs, the method of [28] could be used here. The sleep periods of nodes could make up ISO-counts, the query packet may find the junction of the two targets, and reach the target it interesting through the junction of them.

VI. SIMULATION

We deployed 100 sensor nodes in a 90*90m^2 area, and every node can communicate with its one hop neighbors, as shown in Fig. 8. Blue rings are sensor nodes. The red ring and line are the starting point and trajectory of the target separately.

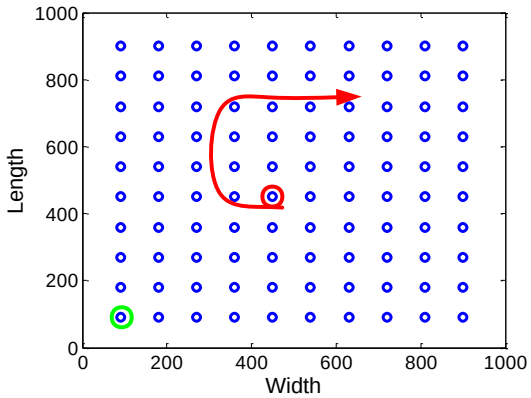


Figure 8. Deployment of simulation

The target stayed at position (450, 450) for 350s which labeled in red ring, and then it moved along the red line with the speed of 1m/s for 720s. Using Theorem 1 $Tu = \frac{D\max}{V\max 2^{(H\max-1)}}$, we can compute out Tu=5s. In this simulation, we set Tc=1s. The node at (90, 90) labeled in green ring is

the user who wants to know the target information.

In initialization phase, the target is at (450, 450), and every node computes out its sleep period as in Fig. 9. When the target stayed at this position for 350s, the numbers of wake up nodes at every second are shown in Fig. 10(a).

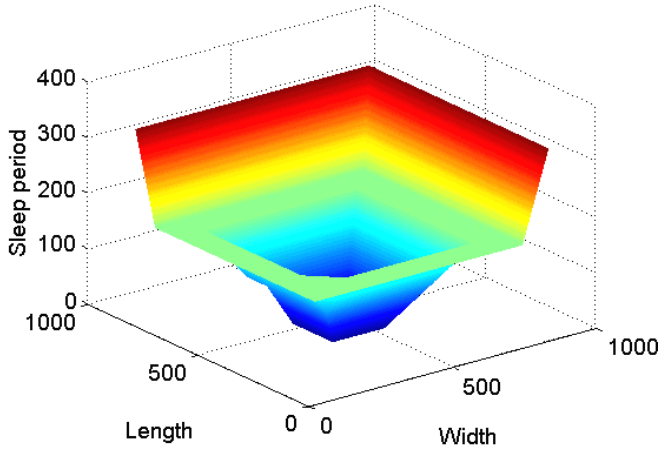
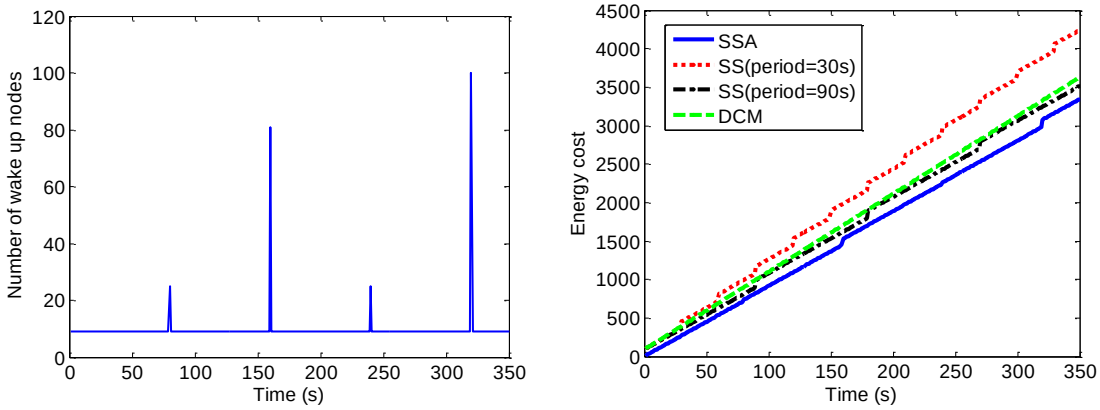


Figure 9. Sleep periods of every node at initialization phase

We compare the energy cost of SSA to "Synchronous Sleep" (SS) and "Dynamic Clusters Mechanism" (DCM) in Fig. 10(b). In SS, all the sleep state nodes' sleep period are equal, thus they sleep at the same time and wake up synchronously. We set sleep periods equal to 30s and 90s. In DCM, nodes near the target make up a dynamic tracking cluster spontaneously to monitor the target, meanwhile other nodes go to sleep with random sleep period within 90s. To simplify computation, we assume the node in sleep state does not cost energy while in all other states costs one unit energy per second. The wake up time of SS and DCM is equal to SSA, which is 1s.

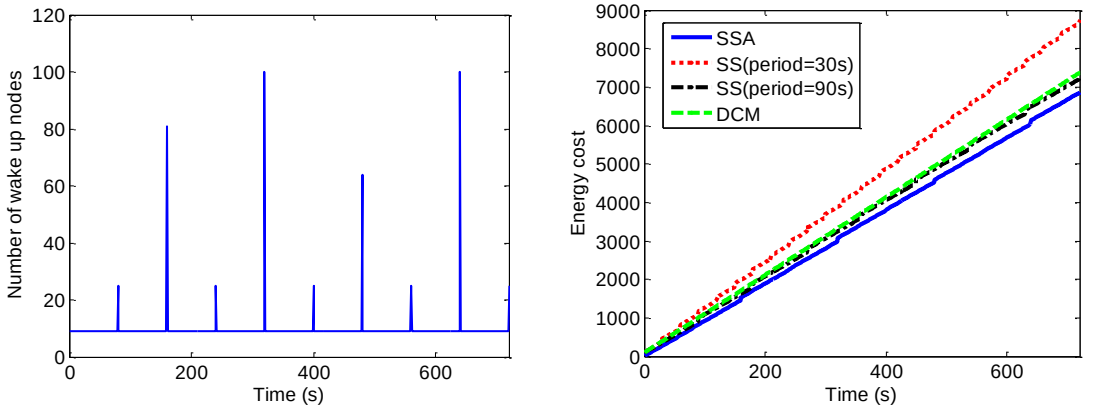
As in Fig. 10(b), the longer the sleep period will cause the less energy cost. But to ensure the target monitoring, the sleep period can't longer than DaveVmax=90s. SSA can save much more energy than SS and DCM, even with the longest sleep period. This is because in SSA the nodes far from the target can sleep long time.



(a) Number of nodes wake up (b) Energy cost

Figure10. Simulation results when the target stationary

When the target was moving, the network turned into state transformation phase. The numbers of wake up nodes at every second are shown in Fig. 11(a), and the energy cost comparison results of SSA, SS and DCM are shown in Fig. 11(b). As in Fig. 11(b), the energy cost of SSA is much smaller than SS and DCM, which is the same reason as in initialization phase.



(a) Number of nodes wake up (b) Energy cost

Figure 11.Simulation results when the target moving

The node at (90, 90) in Fig. 9, labeled in green ring is the user wants to get the target information. It can get information in the initialization phase and state transformation phase when it wakes up. In the state transformation phase, it wakes up at 160s, 320s and 640s. So it can query information at 0s, 160s, 320s and 640s.

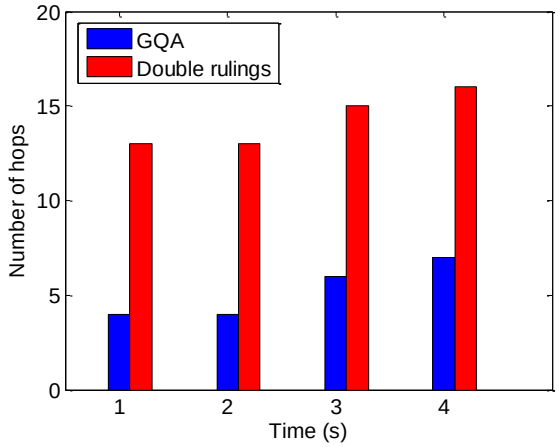


Figure 12. Number of hops needed for the query reaches the target

We compare the route length of GQA to Double Rulings [21] in Fig. 12. The hops of GQA shorter than Double Rulings, which means it costs less energy in query phase. But GQA brings larger latency than Double Rulings, as GQA focuses on the applications of target monitoring in which the target information is not necessarily all the time. The latency of GQA is no longer than $2^{(H\max-1)}T$u, and this latency could be accepted in these applications.

VII. CONCLUSION

For energy constrained target monitoring sensor network systems, duty cycling is an effective technique to prolong network lifetime. In this paper, we propose a novel duty cycling design for the application of target monitoring, which can prolong network lifetime as far as possible, and can respond the target information queries through gradient query routing. Both analytical and experimental results prove the functionality of the proposed design on reduction of energy consumption.

In the further, we will analyze more about the problem of energy cost balance, and study more special issues for our design, such as the case of a target doesn't move for a long period, or a target suddenly enters the monitoring area after the "Initialization phase" and so on.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China under Grant No. 61103196 and 61170258. Research Award Fund for Excellent Young Scientist of Shandong Province under Grant No. BS2012DX011. Qingdao Science and Technology Development Program under Grant No. 12-1-4-3-(16)-jch.

REFERENCES

[1] P. Guo, T. Jiang, Q. Zhang, K. Zhang, "Sleep Scheduling for Critical Event Monitoring in Wireless Sensor Networks", IEEE Transactions on Parallel and Distributed Systems, Volume 23, Issue 2, pp.345-352, 2012.

[2] E. Bulut, I. Korpeoglu, "Sleep Scheduling with Expected Common Coverage in Wireless Sensor Networks", Wireless Networks, volume 17, pp.19-40, 2011.

[3] A. Salam, Hady S., Olariu, Stephan, "Toward Adaptive Sleep Schedules for Balancing Energy Consumption in Wireless Sensor Networks", IEEE Transactions on Computers, Volume 61, Issue 10, pp.1443-1458, 2012.

[4] C. H. Lee, D. Y. Eun, "Smart Sleep: Sleep More to Reduce Delay in Duty-Cycled Wireless Sensor Networks", IEEE INFOCOM, pp.611-615, 2011.

[5] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L .Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, B. Krogh, "Vigilnet: an Integrated Sensor Network System for Energy-Efficient Surveillance", ACM Transaction on Sensor Networks, 2006.

[6] S. Ren, Q. Li, H. N. Wang, X. Chen, and X. Zhang, "Probabilistic Coverage for Object Tracking in Sensor Networks", Mobicom 2004 Poster Session, 2004.

[7] C. Gui, P. Mohapatra, "Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks", ACM Mobicom, 2004.

[8] Y. Guo, Z. Guo, F. Hong, and L. Hong, "Sleep Scheduling and Gradient Query in Sensor Networks for Target Monitoring", NPC, 2009.

[9] C. Liu, K. Wu, V. King, "Randomized Coverage-Preserving Scheduling Schemes for Wireless Sensor Networks", Proceedings of IFIP Networking, 2005.

[10] D. Dong, Y. Liu, K. Liu, X. Liao, "Distributed Coverage in Wireless Ad Hoc and Sensor Networks by Topological Graph Approaches", IEEE 30th International Conference on Distributed Computing Systems (ICDCS), pp.106-115, 2010.

[11] F. Ye, G. Zhong, S. Lu, etal, "A Robust Energy on Serving Protocol for Long Lived Sensor Networks", IEEE Computer Society Press, pp.28-37, 2003.

[12] T. Di, G. Nicolas D, "A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Networks", The 1st ACM International Workshop on Wireless Sensor Network and Application, 2002.

[13] Q. Cao, T. Abdelzaher, T. He, J. Stankovic, "Towards Optimal Sleep Scheduling in Sensor Networks for Rare-Event Detection", Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, 2005.

[14] Bo Sheng, Qun Li, Verifiable "Privacy-Preserving Sensor Network Storage for Range Query", IEEE Transactions on Mobile Computing, Volume 10, Issue 9, pp.1312-1326, 2011.

[15] Omiwade S., R. Zheng, "Maximum Lifetime Data Regeneration for Persistent Storage in Wireless Sensor Networks", IEEE Global Telecommunications Conference (GLOBECOM 2011), pp.1-6, 2011.

[16] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks", Mobi-Com 2000: Proceedings of the 6th Annual International Conference on Mobile Computing And Networking, pp.56-67, 2000.

[17] S. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong, "Tag: a Tiny Aggregation Service for Ad-Hoc Sensor Networks", Proceedings of the 5th Symposium on Operating Systems Design and Implementation, pp.131-146, 2002.

[18] S. Ratnasamy, L. Yin, F. Yu, D. Estrin, R. Govindan, B. Karp, and S. Shenker, "GHT: A Geographic Hash Table for Data Centric Storage in Sensornets", Proc. of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), pp.78-87, September 2002.

[19] S. Funke, L. Guibas, A. Nguyen, and Y. Wang, "Distance-sensitive Routing and Information Brokerage in Sensor Networks", DCOSS 2006, pp. 234-251, 2006.

[20] X. Li, Y. J. Kim, R. Govindan, W. Hong, "Dimensional Range Queries in Sensor Networks", Proceedings of the first International Conference on Embedded Networked Sensor Systems, pp.63-75, 2003.

[21] R. Sarkar, X. Zhu, J. Gao, "Double Rulings for Information Brokerage in Sensor Networks", MobiCom06, 2006.

[22] X. Liu, S. Zhou, G. Bai, D. Zhu, "Multidimensional Similarity In-network Query for Large-Scale Sensor Networks", Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, pp.305-310, 2009.

[23] J. Zheng, W. Jia, G. Wang, . Wu, "Target Trajectory Querying in Wireless Sensor Networks", IEEE International Conference on Communications (ICC), pp.1-6, 2010.

[24] J. Faruque, K. Psounis, and A. Helmy, "Analysis of Gradient Based Routing Protocols in Sensor Networks", IEEE/ACM Int'l Conference on Distributed Computing in Sensor Systems (DCOSS), pp.258-275, June 2005.

[25] Garg, R.,Varna, A.L., MinWu, "An Efficient Gradient Descent Approach to Secure Localization in Resource Constrained Wireless Sensor Networks", IEEE Transactions on Information Forensics and Security, Volume:7, Is-sue:2, pp.717-730, 2012.

[26] E. Ochir, O., Minier, M.,Valois, F., Kountouris, A., "Toward Resilient Routing in Wireless Sensor Networks: Gradient-Based Routing in Focus", International Conference on Sensor Technologies and Applications (SENSORCOMM), pp.478-483, 2010.

[27] H. Lin, M. Lu, N. Milosavljevic, J. Gao, L. J. Guibas, "Composable Information Gradients in Wireless Sensor Networks", International Conference on Information Processing in Sensor Networks (IPSN 2008), pp.121-132, 2008.

[28] R. Sarkar, X. Zhu, J. Gao, L. J. Guibas, J. S. B. Mitchell, "Iso-Contour Queries and Gradient Descent with Guaranteed Delivery in Sensor Networks", 27th Annual IEEE Conference on Computer Communications (INFOCOM'08), May, 2008.

[29] M. Lei, X. Bu-gong, "Energy Optimization Strategy for Target Tracking in Wireless Sensor Networks", Journal of South China University of Technology, 2011, 39(7), 13-20