Computing (2014) 96:995–1010
DOI 10.1007/s00607-013-0364-7

RENDEZVOUS: towards fast event detecting in wireless sensor and actor networks

Mianxiong Dong · Kaoru Ota · He Li · Suguo Du · Haojin Zhu · Song Guo

Received: 28 February 2013 / Accepted: 11 October 2013 / Published online: 24 October 2013

Abstract Fast event detection is important in wireless sensor and actor networks (WSANs) since actors can perform appropriate actions which sensor nodes are not capable to do. While WSANs inherits the typical constrains of WSNs such as energy and computation limitations of sensor nodes. In this paper, we propose a fast event detecting algorithm named RENDEZVOUS to accelerate the actor's event detecting process while keep the energy consumption of sensor nodes as minimum. When design RENDEZVOUS, we first study the mobility control of a actor to help the actor move around close to a event by using Reinforcement Learning techniques with collected sensory data. We then design a scheme to search nearby actors from the event side inspired by a searching behavior of desert ants. By both perform search actions from sensor side and actor side, the proposed algorithm can achieve fast event detecting with neglect-able additional energy cost on sensors side. Extensive simulation results demonstrate the efficiency of RENDEZVOUS.

M. Dong (✉) · H. Li · S. Guo
University of Aizu, Aizu-Wakamatsu, Japan
e-mail: mx.dong@ieee.org

H. Li
e-mail: d8141105@u-aizu.ac.jp

S. Guo
e-mail: sguo@u-aizu.ac.jp

K. Ota
Muroran Institute of Technology, Muroran, Japan
e-mail: ota@csse.muroran-it.ac.jp

S. Du · H. Zhu
Shanghai Jiao Tong University, Shanghai, China
e-mail: sgdu@sjtu.edu.cn

H. Zhu
e-mail: zhu-hj@sjtu.edu.cn

Keywords Wireless sensor and actor networks · Mobility control · Event detection · Energy efficiency

Mathematics Subject Classification 68M10

1 Introduction

Recent advance in low-power wireless communications and micro-electronics technology have witnessed a rapid penetration of wireless networks [1–6] in our daily life. For a large-scale wireless sensor network, efficiently monitoring environments and processing sensory data is necessary for conserving both lifetime and energy of the network [7]. Wireless sensor and actor networks (WSANs) successfully accomplishes this requirement to collect, process data from sensors and perform appropriate actions in the network [8,9]. In WSANs, sensors are low-cost and low-power and are deployed throughout a field to sense environments, while actors are powerful and resourceful and are deployed much less than the number of the sensors. The actors collect and process data reported by the sensors and perform actions according to situations when the sensors detect events in the monitored field. In general, the application scenarios in WSANs can be categorized basically into two types: complement of existing sensor networks and new application paradigms. The first category includes application scenarios where actors are employed for complementing/enhancing traditional sensor networks. The other category includes application scenarios where actors are employed in a new way such that actors react to events and work for "mission" accomplishment in event areas. For example, when a disaster happens (e.g., earthquake) and some people are missing, sensors find locations of victims and accordingly actors guide a rescue team to the victims along safe routes [10]. Also, WSANs could deployed at mountain area to monitor forest for wildfire. If a fire is detected by a sensor node, an actor can quickly go to the event and perform fire extinction. In this paper, we focus on the new application paradigms and assume an action area of actors coincides with an event area or is very close to the event area.

In literature, actors always walk randomly around the network before they detect events from sensory data. Such kind of movement may cause not only data delivery delay but also energy consumption on sensors if the location of the actors is far from event areas which results in consuming enormous amounts of energy over the network. Also, if actors take quite long time to migrate to action areas due to the long distance between the action areas and the location of the actors, such a waste of time could inflict considerable damage on mission-critical applications. An intuitive solution is to let sensors periodically report sensory data so that actors can arrive at event areas just on time when events occur. However, it might cause prohibitive network energy and bandwidth consumption. Such problems can be solved by elaborating the actor's mobility pattern which makes actors move close to areas before events actually occur. Hence, our objective is to design an effective and efficient algorithm to fast detect the event by a actor, especially in case of emergent situations, while saving energy on sensors and time for the actor to migrate after events.

In this paper, we propose an innovative fast event detecting scheme named RENDEZVOUS which elegantly integrate the detecting process from both actor side and sensor side. The design of RENDEZVOUS are based on two parts. We first design a scheme to control actor's mobility which can formulate actor's movement around the event area by using Reinforcement Learning Techniques. We then design the searching algorithm from sensor side to actively find the actor near the event. Our searching scheme is inspired by a desert ant searching it's nest in nature scenario. When the actor "meets" the searching query, it can directly go to the event area without calculating the rote path by collecting the sensory data from sensors. Extensive simulation demonstrate the efficacy of RENDEZVOUS.

The remainder of this paper is organized as follows. In Sect. 2, we present related work and the network model considered throughout the paper. In Sect. 3, we design the actor's mobility control and give the preliminary result. We design searching algorithm of sensor side in Sect. 4. The simulation results of RENDEZVOUS are given in Sect. 5, followed by the conclusion in Sect. 6.

2 Related work and network model

2.1 Related work

Research issues on WSANs have been widely addressed by many articles recently [8–11]. Selvaradjou et al. [11] formulate the problem of optimal assignment of mobile actors in WSANs as Mixed Integer Non Linear Program to conserve the energy needed for actor mobility but otherwise fulfill the deadline and resource constraints of events. Akyildiz et al. [8] introduce lots of open research challenges for sensor–actor and actor–actor coordination and communication problems in WSANs. Melodia et al. [10] handle the coordination problem in WSANs and propose a location management scheme to localize actors with minimal energy expenditure for sensors and design algorithms to deal with modeling actors' mobility for optimal task accomplishment. Ota et al. [12] propose ORACLE to make actors predict events before sensors detection and migrate to the areas where the event may occur. Dong et al. [13,14] propose HERVEST to collect data with an application-oriented Mobile Actor (MA) based on the uncertainty of sensory data provided by all n-hop neighbor nodes. Martirosyan et al. [15] present the performance evaluation of an algorithm for preserving temporal relationships of events in wireless sensor actor networks (WSANs). The goal of the proposed event ordering algorithm for WSANs is to reduce the overhead in terms of energy dissipation and delay. However, those papers only consider the actors' motion while less paying attention on the sensor side. To the best of our knowledge, this is the first work to address the fast event detecting problem by elegantly integrate the actor's mobility control and sensor's searching procedure.

2.2 Network model

A typical WSAN is composed of a large number of sensors and actors in the network area. In a certain field, the actor moves around the network from the beginning of surveillance and collects sensory data from every sensor encountered. On the other

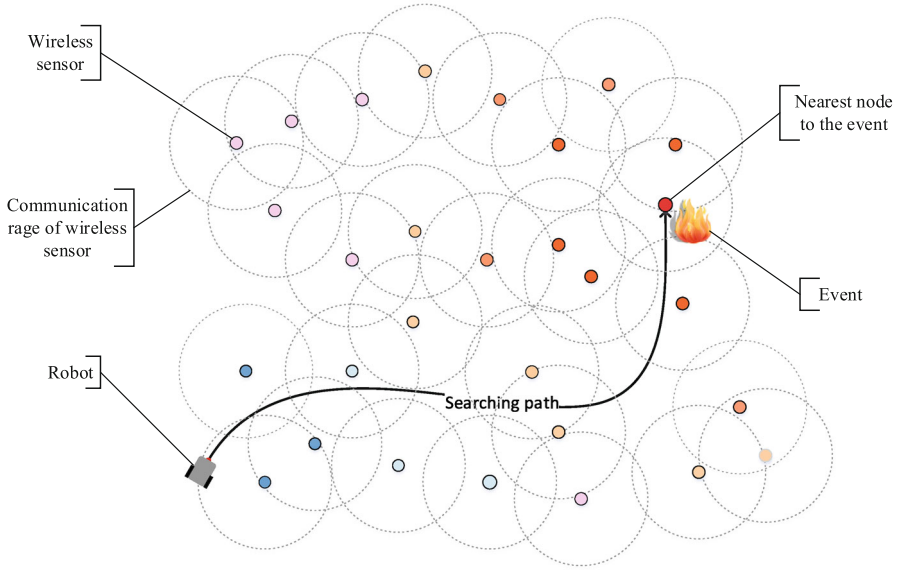

Fig. 1 The network model of WSANs

hand, the sensors are densely deployed on a sensing field and monitor environmental phenomenon and periodically store sensory data on themselves. In the deployment stage, the sensors location is known by the actors. Each sensor's communication range is very limited and the sensors can communicate only with their neighbors or actors inside their communication range. A sensor can also transmit data directly to their neighbor or the actor. It has a limited memory to store information with battery powered. When an event happen, the sensor detected the event and actor will go to the event area to perform some missions. Figure 1 shows the network mode of a WSAN in our paper. An event (fire) is detected by the sensor node around it. Then the actor (Firefighting robot)'s mission is to find the event and to put out the fire.

3 RENDEZVOUS: actor mobility control scheme

3.1 Definition of direction

It is important to effectively select the direction of sensors to be visited since unnecessary visiting results in consuming both energy on sensors and time for the actor to move. Thus, we define a direction as one of neighboring nodes where the actor migrates at the next time slot. The actor selects one node based on sensory data of each neighbor as criteria to make a decision. For the actor to decide the direction without finding specific sensors, we define a feature of each sensor, called the *propensity* of a sensor, to show the sensor's surrounding information by involving its neighboring nodes. The *propensity* quantifies information produced by a sensor and its neighbors and shows how diverse data the information includes. For example, if sensors observe temperature of an environment, the *propensity* of a sensor is low when the sensor and its neighbors all produce the similar temperature values.

For calculating the *propensity* on each sensor, when an actor moves into a communication range with sensors, each sensor in the range will send its own sensory data to the actor then the *propensity* will be calculated on the actor side with all the sensory data.The *propensity* is recalculated when new data comes in another period of time in order to keep temporal correlation among sensory data.

The *propensity* is computed using entropy as follows. Let X be the random variable representing a certain type of sensory data (e.g., temperature) from sensor i and its neighbors. For the sake of computation, we discretize the continuous sensory data values into Q disjoint intervals. If we have observed the sensors for N time slots in a period of time ΔT, the time series of the sensory data can be denoted by $D_i = (d_0, d_1, \ldots, d_{N-1})$ where $d_t \in [0, Q-1], 0 \leq i \leq N-1$ is the sensory data in time slot t. Assume each of these Q data values appeared m_v times in $D_i, 0 \leq v \leq Q-1$. Thus, the probability of the data value on the nodes being equal to particular value v can be computed as $\frac{m_v}{N}$. Therefore, the entropy of X which is denoted as the *propensity* and the *propensity* of node i at time t can be described as

$$propensity_i(t) = H(X) = -\sum_{v=0}^{Q-1} \frac{m_v}{N} \log \frac{m_v}{N} \tag{1}$$

3.2 Reinforcement learning in Markov decision processes

We design actor mobility control using reinforcement learning in MDPs which is a well-known mathematical framework to dynamically formulate optimization problems of stochastic systems such as a queuing model and an inventory model. The MDP also has been widely adopted for sensor network research [16,17]. A ferry's mobility control framework in mobile ad hoc networks is designed based on an extended MDP called Partially Observable MDP (POMDP) [17], which aims for a data ferry to encounter as many as randomly moving sensors by controlling the ferry's motion using the POMDP. Our research is inspired by this research but we apply the MDP instead of the POMDP since the actor can directly observe the current state by communication between the actor and deployed sensor nodes.

The MDP is represented by a four-tuple as shown in Table 1. The main objective is to find a policy π for the actor to migrate to next direction (action) under the current state. π is a function specifying the action to be selected when the current state is s. Whenever the actor selects an action and then a state is changed, the actor obtains a reward $r(s, a)$ based on given state s and action a. We define evaluation function $f(X_t)$ for the actor to evaluate the current state where X_t is some sensory data collected from nodes within the actor's communication range at current time slot t. The actor compares the output of $f(X_t)$ to the output of $f(X_{t-1})$ calculated at previous time slot $t-1$, and then evaluates current state s. If $f(X_t) \geq f(X_{t-1})$, s is set as *Increase*, otherwise $s = decrease$. We assume the evaluation function depends on applications such that what kind of events is targeted to be detected in the monitoring field. However, the evaluation function should be designed consistently in principle with rewarding process. Briefly, the greater the output of the evaluation function, the better the current

Table 1 A four-tuple $(S, A, P(\cdot,\cdot), r(\cdot,\cdot))$ in the MDP

S	A set of state s_t denotes an increase or a decrease in the output of evaluation function $f(X_t)$ to the previous one $f(X_{t-1})$, s.t. $S := \{increase, decrease\}$
A	A set of actions and action a_t denotes one of neighboring sensors to migrate at current time step t. Based on the propensity of the neighboring sensors, A includes three sensors whose *propensity* is the highest, medium, and the lowest in all neighbors respectively s.t. $A := \{j_h, j_m, j_l\}$
$P(\cdot,\cdot)$	The state transition function represented by $P_a(s, s') = Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ which is the probability of state s' at next time step $t+1$ if the actor chooses action a in state s at time step t
$r(\cdot,\cdot)$	The immediate reward represented by $r(s, a)$ which is the reward the actor obtains when the actor takes action a after that it observes state s. We set $r(s, a) = 1$ when $s = increase$ otherwise $r(s, a) = -1$

state and the actor is rewarded. We will show an example of the evaluation function used in our simulations in Sect. 3.3.

The goal is to choose a policy π that will maximize some cumulative functions of the rewards which can be denoted by $R(s_T, a_T)$ where s_T and a_T are a history of states and actions over a potentially long time window T respectively. That is formulated as:

$$\pi^* = argmax E[R(s_T, a_T) | a_t = \pi_t(s_{t-1}, a_{t-1})] \tag{2}$$

Although the actor's mobility is modeled as Markovian such that it decides the next sensor to migrate only depending on a current state, the actor does not know the exact probabilities of the state transition. Then, the problem becomes a reinforcement learning problem in the MDP [18]. A basic idea of the reinforcement learning is the actor repeats trials of taking several actions with several states and figures out an optimal policy from the experience. For effective learning of the actor, a state-value function is defined to evaluate each state where the actor expects to be given rewards afterwords:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}(r_{t+1} + \gamma r_{t+1} + \ldots | s_t = s) \\ &= E_{\pi}\left(\sum_{k=1}^{\inf} \gamma^{k-1} r_{t+k+1}\right) \end{aligned} \tag{3}$$

where γ is the discount rate $(0 < \gamma \leq 1)$ to evaluate a reward given in the future which is discounted over time because a farther-future reward is less guaranteed since the environment surrounding the actor may change over time and the actor may not get a constant reward. In our scenario, the actor considers not only the impact of the state, also the impact of an action selected when in the current state on the rewards. For this purpose, an action-value function is defined to evaluate a pair of a state and an action:

$$Q^{\pi}(s,a) = E_{\pi}(r_{t+1} + \gamma r_{t+1} + \ldots | s_t = s, a_t = a)$$
$$= r(s,a) + \gamma \sum_{s'} V^{\pi}(s') P_a(s,s') \quad (4)$$

An optimal action-value function which satisfies the above action-value function for every pair of a state and an action is described as:

$$Q^*(s,a) = \max \left[r(s,a) + \gamma \sum_{s'} Q^*(s',a) \right] \quad (5)$$

The policy $\pi(s)$ can be determined by the optimal action-value function Q^*, which is a precise estimate of the action-value function Q^{π}. Then, the question is how to update a Q-value during the actor's experience in order to finally obtain the optimal-value function Q^*. We apply a reinforcement learning technique called Q-learning to the problem [18]. The basic idea of Q-learning can be described as follows. When the actor takes an available action a_t in state s_t, Q-value $Q(s_t, a_t)$ is updated with max $Q(s_{t+1}, a_{t+1})$ where the actor is expected to choose action a_{t+1} which has the maximum Q-value in state s_{t+1}. The Q-value converges to an optimal Q-value with probability one if every action is chosen by the actor a number of times although these actions are chosen in a random way. However, the actor is expected to acquire as many as rewards before obtaining the optimal Q-value in order to get close to a possible event area. Therefore, we apply the mostly used ϵ-greedy action-selection rule which is, an action with a maximum Q-value at time step t is selected while the other actions are randomly selected with small probability ϵ. This can utilize estimated Q-values obtained from learning and at the same time efficiently seek better solutions for future. Algorithm 1 summarizes procedures of our proposed method of actor mobility control.

Algorithm 1 Mobility control

```
Initialize Q-value;
while Find a node which knows a location of a node detecting an event do
  Broadcast a request of sending its historical sensory data to each node in the communication range of
  the actor;
  Using latest sensory data of each node, calculate evaluation function value f(X_t) ;
  Update current state s_t and get reward r_{t-1};
  if Q-value is optimal then
    Set action selection rule F as the optimal function Q*;
  else
    Update Q-value;
    Set action selection rule F as ε-greedy method;
  end if
  Calculate Entropy of each node based on the historical sensory data;
  Update a set of actions A;
  Select one of the nodes by method F;
  Shift a time step from t to t + 1;
  Migrate to the selected node;
end while
```

Table 2 Parameters and value setting in the experiments

Parameter	Value setting
Monitored area	Small-sized: $80 \times 80\,\mathrm{m}^2$
	Large-sized: $160 \times 160\,\mathrm{m}^2$
Transmission range of sensor and actor	10 m
ϵ for ϵ-greedy action-selection rule	0.1

3.3 Preliminary result of RENDEZVOUS

We evaluate our proposed scheme by simulation experiments using Netlogo simulator [19]. In our simulation, we consider one actor take in charge of a monitored area and one event occur in the area. Two key performance metrics in the experiments distance and range are evaluated.

- The distance is between the actor and a sensor node which first detects an event. It is shorter when the actor is closer to an event area, which implies the actor successfully predicts the event.
- The range indicates a moving range of the actor in the monitored area. It is ideal for the range to be minimized while the shorter distance is maintained.

In our network settings, a number of sensors are randomly and densely deployed. The event area is randomly chosen within the monitored area and environmental values are set on each sensor, which are spatially-correlated to the event area and change every unit of time. We randomly generate 50 network-examples for each network setting to obtain the distance and the energy consumption respectively from the average over those examples. We set the monitored area small-sized and large-sized: $80 \times 80\,\mathrm{m}^2$ and $160 \times 160\,\mathrm{m}^2$, respectively. A transmission range of sensors is 10 m in each set of experiments. We set $\epsilon = 0.1$ for the probability used in the ϵ-greedy action-selection rule. Those parameters are summarized in Table 2.

In this simulation, we use the following evaluation function:

$$f(X_t) = -\sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}(th - x_i)^2} \tag{6}$$

where n is the number of sensor nodes within the actor's communication range, $X_t = (x_1, x_2, \ldots, x_n)$ is those nodes' sensory data collected at time slot t, and th is a constant and indicates the threshold value when a sensor node detects the event occurs.

We compare our method with a random walk model where the actor randomly move around from sensor to sensor until the actor receives any event report from a sensor node.

Figure 2 shows CDF of the distance from the event area to the actor in the small-sized area by using the random model and ours, respectively. The actor using our method more successfully move close to the event area than using the random walk method while the actor observes the same range of the area by using either our method or the random walk model as shown in Fig. 3. The results imply that, with our proposed

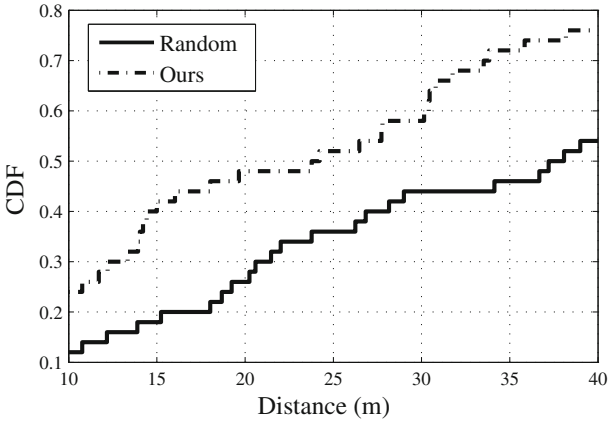

Fig. 2 CDF of the distance between the actor and a sensor node detecting the event in a small-sized area

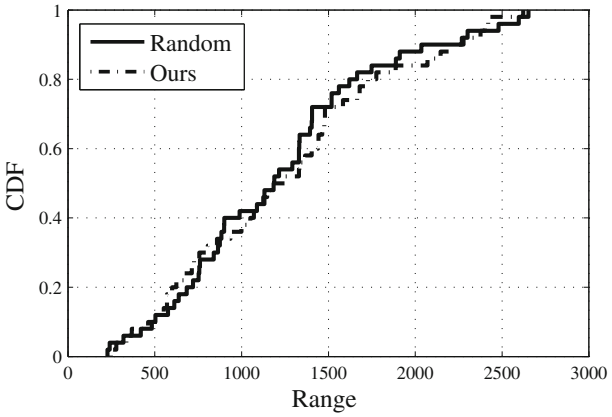

Fig. 3 CDF of the range where the actor migrates in a small-sized area

scheme, the actor effectively observes the monitored area and properly finds a clue in search for the possible event from local observations.

According to Figs. 4 and 5, results for the large-sized area are similar to ones for the small-sized area. However, the actor with our method can avoid the worst case when it moves in the largest range of the area by using the random walk method as shown in Fig. 5. This implies that the actor can minimize its motion when the monitored area is large.

4 RENDEZVOUS: sensor seeks for actor

4.1 Algorithm overview

In the preliminary result, we find our proposed scheme outperform than the random walk. While we also find an interesting observation that is when an actor performs the event detection with Reinforcement Learning, the closer the actor to the event, the

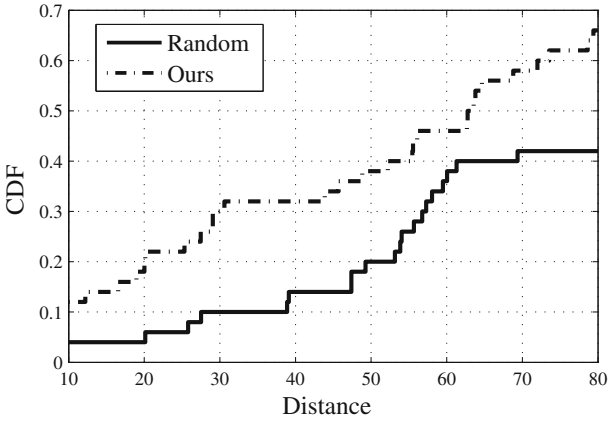

Fig. 4 CDF of the distance between the actor and a sensor node detecting the event in a large-sized area

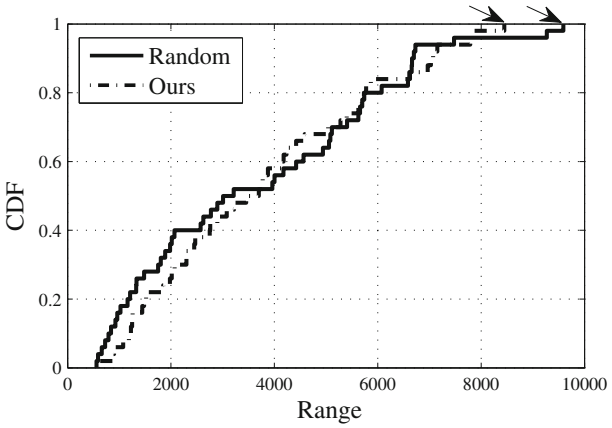

Fig. 5 CDF of the range where the actor migrates in a large-sized area

decision to next direction is more difficult to find. Sometimes the actor could take a long time to find the event even it is very close to the event. This is because the difference of the data becomes small around the event area. For example, if a fire happened, the temperature around the fire could be very similar compare the place far from it. We find the mobility control is like a macroscopic view of the event detection procedure. Now we need a precise microscopic view of when an actor is close to the event. One way to expand the information of an event is to broadcast the current sensor's information though the whole network [20]. However, due to the limited energy of sensor node, the flooding algorithm generate prohibitive network traffic which results the significant energy consumption and finally make a dead area *hole* of the sensor network. To overcome these limitations, therefore to fast deliver the event to the actor near around, we design the sensor part of RENDEZVOUS, to let both the actor and sensor try to meet each other in an meeting area close to the event. RENDEZVOUS is divided to two parts. The first part is Actor seeks for sensor. We use the Reinforce Leaning algorithm to control the actor's mobility and let the actor move towards the

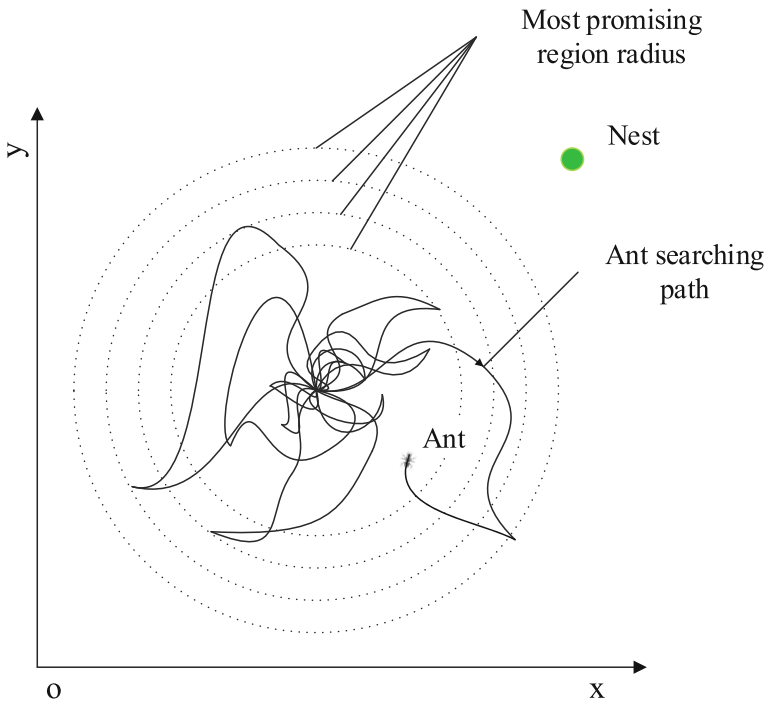

Fig. 6 An example of search pattern of a desert ant

event area which already described in the above section. The second part is sensor seeks for actor. When actor comes to the region near the event, we try to set a ”space” which can help to actor and sensor can meet.

The idea of seeking actor of RENDEZVOUS is inspired a nature algorithm [21] from the mobility patten of a desert ant seeks for its nest. If an ant get lost, it will not perform a random walk search of its nest but excuse a number of loops search with ever-increasing way. The ant's searching pattern can be summarized as following. When the ant get lost, it tries to seek the nest with a believe that the nest is near its current location. So the place near the center is most intensively be searched. This is perfectly matching to the situation we discussed above, i.e. an event (ant) is near to an actor (nest). Figure 6 shows an example of ant's mobility of the searching algorithm. An ant starts searching from the origin with a ever-increasing loop. At each loop searching, the ant will first increase the distance from the origin. If the distance becomes larger than the radius of the current loop, the ant will perform a searching back to the origin.

4.2 Algorithm design

In this section, we discuss the detail design of the searching scheme. Following the essence of the ant search strategy, when a event happen, the sensor will detect event and try to find the actor near this region. We define the sensor which detected the event as a *source* node of this search. The search process include two directions. The first one is called *outboundsearch* which means the search direction is from the *source*.

We also define the *inboundsearch* with the search direction is towards the *source*. RENDEZVOUS starts with a parameter h which denotes how many hops the algorithm will conduct search in one step. The *source* will first randomly pick one neighbor node to search the actor [21,23]. If the actor is not in the communication range, and the current hop number is small than h, the algorithm is still in the *outboundsearch* phase. The current node will pass the search query to its next neighbor whose hop number is larger than the current node to *source*.

Algorithm 2 Ant search

Input: Node t received a message $msg(direction, ant\text{-}msg, count\text{-}hop, TTL)$;
if t find a actor **then**
  $count\text{-}hop \leftarrow 0$;
  Send a result message to the event node;
else
  if $msg.TTL \geq 0$ **then**
    compute the distance between node nd and the origin node;
    if $msg.direction == outward$ **then**
      if $count\text{-}hop > 0$ **then**
        randomly select a neighbor nodes me which has farther distance from the origin;
        create a massage $msg'(direction, ant\text{-}msg, count\text{-}hop - 1, TTL - 1)$;
        send msg' to node me;
        if $count\text{-}hop == 1$ **then**
          Store the location of origin;
        end if
      end if
    else
      randomly select a neighbor nodes me which has nearer distance from the origin;
      create a meeeage $msg'(inward, ant\text{-}msg, 0, TTL - 1)$;
      send msg' to node me;
    end if
    if $msg.direction == inward$ **then**
      if $t == msg.origin$ **then**
        randomly select a neighbor nodes me;
        create a massage $msg'(direction, ant\text{-}msg, out\text{-}hop + 1, TTL - 1)$;
        send msg' to node me;
      end if
    else
      randomly select a neighbor nodes me which has nearer distance from the origin;
      create a meeeage $msg'(inward, ant\text{-}msg, 0, TTL - 1)$;
      send msg' to node me;
    end if
  end if
end if

5 Performance evaluation

We use the same setting mentioned in Sect. 3.3, except for the range of the monitored area and performance metrics. In this experiments, the monitored area is as $200 \times 200\,\mathrm{m}^2$ and the actor's transit speed is 6 kmph. We consider the following three performance metrics:

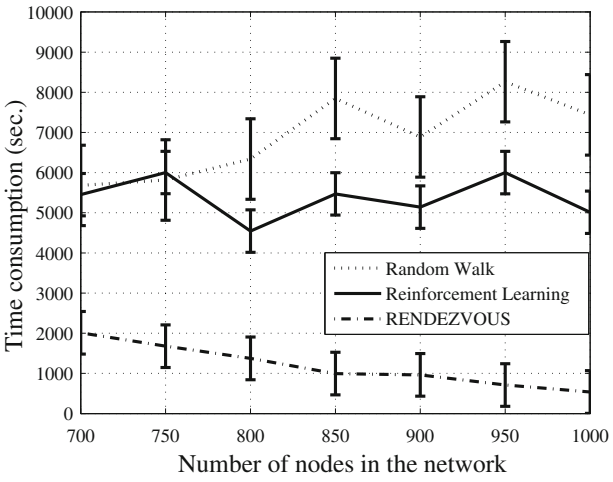

Fig. 7 Time consumption

- Time consumption: Time period until when the actor finds and migrate to a sensor which detected an event.
- Cache: How many sensor nodes are used to store ant messages in the outbound search phase when using RENDEZVOUS. It is represented by a ratio of the number of nodes storing the message to the total number of nodes in the network.
- Energy consumption: Sensor energy consumed while searching the actor. It is calculated using the link metric [22]: $E = 2E_{elec} + E_{amp}d^{\alpha}$ where α is the exponent of the path loss propagation, E_{amp} is a constant, and E_{elec} is the energy for the transceiver circuitry to transmit or receive one bit. $\alpha = 2$, $E_{amp} = 10p$, and $E_{elec} = 50n$ are used in this experiment.

First, we evaluate performance of our RENDEZVOUS in terms of time consumption. We compare time consumption when the actor randomly walks until it finds an event area (referred to as *Random Walk* hereafter) and when the actor purely uses the mobility control based on Reinforcement Learning to find an event area (referred to as *Reinforcement Learning* hereafter), to time consumption when the actor and sensors use RENDEZVOUS.

Figure 7 shows the time consumption of actor migration with Random Walk, Reinforcement Learning, RENDEZVOUS, respectively in a different size of networks including 700–1,000 nodes. As we can see the results, RENDEZVOUS always outperforms other two methods. We conclude that the actor is able to find an event area in less time by using our proposed algorithm.

Second, we evaluate the cache using RENDEZVOUS when changing the total number of sensor nodes in the network. Figure 8 is showing that even introducing ant mimic searching, the cache space is required as an acceptable level.

Lastly, we evaluate the energy consumption of the network by varying the number of sensor nodes. In addition to Reinforcement Learning, we also compare the energy consumption when the network uses a broadcast strategy (hereafter referred to as *Broadcast*) to notify the actor of a location of the event area. As shown in Fig. 9,

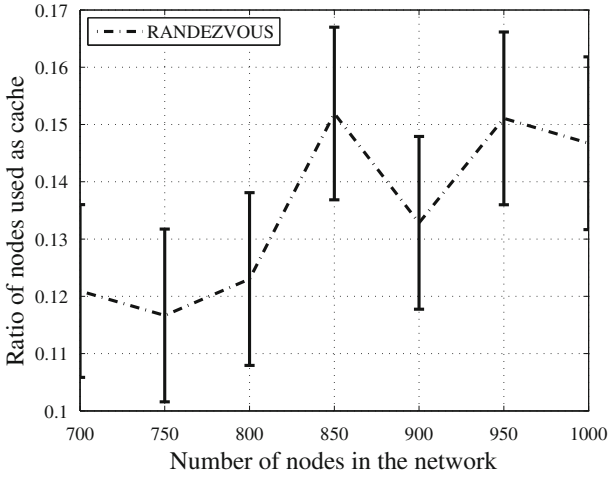

Fig. 8 Cache when use RENDEZVOUS

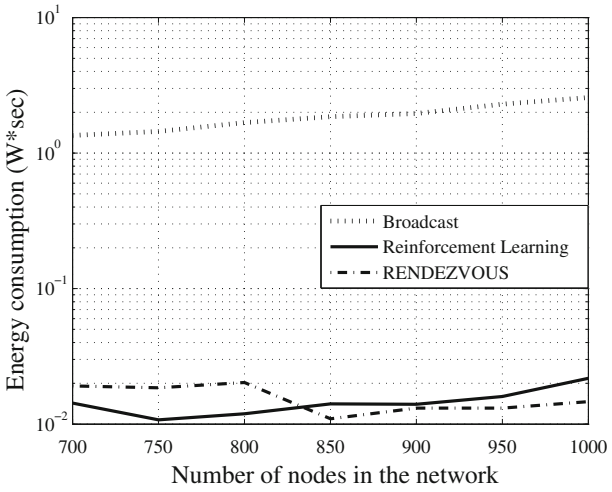

Fig. 9 Energy consumption

the energy consumption of Broadcast is much larger than other two because all sensor nodes in the network get involved to pass packets to inform the event location originating from a sensor node which detected the event. On the other hand, RENDEZVOUS only requires to use a part of sensor nodes in the vicinity of the event area and the energy consumption induced by RENDEZVOUS is neglectable. The results indicate that the sensor energy can be saved while satisfying the actor to take less time to transit to the event area.

We also evaluate the energy dispersion of the sensor nodes in each method. This metric is one of the most important factors for sensor networks since unbalanced use of energy results in emerging unobservable areas in the network and narrowing the network coverage. Figure 10 shows the standard deviation (SD) of the energy consumption

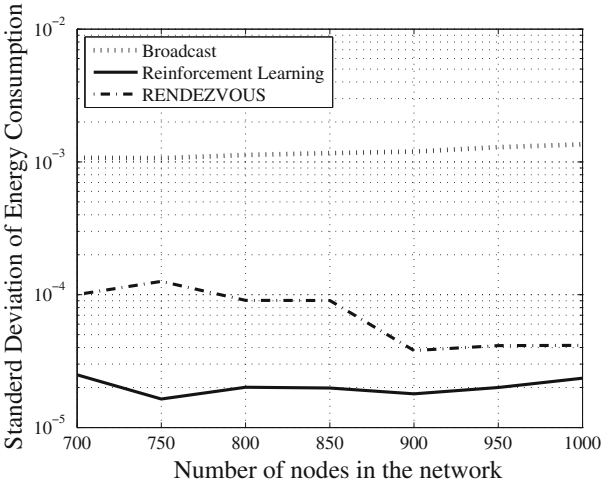

Fig. 10 CDF of the range where the actor migrates in a large-sized area

of each node in the whole network. As we can see, both Reinforcement Learning and RENDEZVOUS effectively distribute energy in the network comparing to Broadcast. Performance of Reinforcement Learning is slightly better than RENDEZVOUS. The reason of that is some sensor nodes close to the event area consume more energy than others because they get involved in delivering ant messages in RENDEZVOUS. Meanwhile, in Reinforcement Learning, each sensor node consumes energy more evenly since the actor communicates with sensor nodes on a route path of its migration. However, according to the result shown in Fig. 9, the SD of the energy consumption in RENDEZVOUS is still acceptable and leads us to conclude that RENDEZVOUS is enough effective for energy dispersion in the network.

6 Conclusion and future work

In this paper, we have proposed RENDEZVOUS for fast event detecting in Wireless Sensor and Actor Networks. Extensive simulation analysis demonstrated the performance of our proposed scheme that achieving both time- and energy-effectiveness. For our future work, we will consider multi-actors' mobility control to deal with several events occurring simultaneously regarding velocity of each actor, *propensity* of each event, and mutual correlation of the both metrics.

Acknowledgments This work is partially supported by JSPS KAKENHI Grant Number 25880002, JSPS A3 Foresight Program, NEC C&C Foundation, National Science Foundation of China (Grant No. 70971086, 61003218, 61272444, 61161140320, 61033014, 71061005), and Doctoral Fund of Ministry of Education of China (Grant No. 20100073120065).

References

1. Fernandes S, Karmouch A (2012) Vertical mobility management architectures in wireless networks: a comprehensive survey and future directions. IEEE Commun Surv Tutorials 14(1):45–63

2. Paul B, Marcombes S, David A, Struijk LNA, Le Moullec Y (2013) A context-aware user interface for wireless personal-area network assistive environments. Wireless Personal Communications, pp 1–21
3. Wu J, Dong M, Ota K, Zhou Z, Duan B (2013) Towards fault-tolerant fine-grained data access control for smart grid. Wireless Personal Communications, pp 1–22, 2013
4. Wu Y, Min G, Al-Dubai A (2012) A new analytical model for multi-hop cognitive radio networks. IEEE Trans Wireless Commun 11(5):1643–1648
5. Min G, Wu Y, Al-Dubai A (2012) Performance modelling and analysis of cognitive mesh networks. IEEE Trans Commun 60(6):1474–1478
6. Zhu H, Dong M, Chang S, Zhu Y, Li M, Shen X (2013) Zoom: scaling the mobility for fast opportunistic forwarding in vehicular networks. In: Proceedings of IEEE INFOCOM, 2013, pp 2832–2840
7. Ota K, Dong M, Wang J, Guo S, Cheng Z, Guo M (2010) Dynamic itinerary planning for mobile agents with a content-specific approach in wireless sensor networks. In: Proceedings of 2010 IEEE 72nd vehicular technology conference, pp 1–5
8. Akyildiz IF, Kasimoglu IH (2004) Wireless sensor and actor networks: research challenges. Ad Hoc Netw 2(4):351–367
9. Chen J, Yu Q, Cheng P, Sun Y, Fan Y, Shen X (2011) Game theoretical approach for channel allocation in wireless sensor and actuator networks. IEEE Trans Autom Control 56(10):2332–2344
10. Melodia T, Pompili D, Akyldiz IF (2010) Handling mobility in wireless sensor and actor networks. IEEE Trans Mobile Comput 9:160–173
11. Selvaradjou K, Dhanaraj M, Murthy CSR (2006) Energy efficient assignment of events in wireless sensor and mobile actor networks. In: 14th IEEE international conference on networks, 2006. ICON '06, vol 2, pp 1–6
12. Ota K, Dong M, Cheng Z, Wang J, Li X, Shen XS (2012) Oracle: mobility control in wireless sensor and actor networks. Comput Commun 35(9):1029–1037
13. Dong M, Ota K, Li X, Shen XS, Guo S, Guo M (2011) Harvest: a task-objective efficient data collection scheme in wireless sensor and actor networks. Int Conf Commun Mobile Comput 0:485–488
14. Ota K, Dong M, Li X (2009) Tinybee: mobile-agent-based data gathering system in wireless sensor networks. Int Conf Netw Archit Storage 0:24–31
15. Martirosyan A, Boukerche A (2012) Preserving temporal relationships of events for wireless sensor actor networks. IEEE Trans Comput 61(8):1203–1216
16. Yeow W-L, Tham C-K, Wong W-C (2007) Energy efficient multiple target tracking in wireless sensor networks. IEEE Trans Veh Technol 56(2):918–928
17. He T, Lee K-W, Swami A (2010) Flying in the dark: controlling autonomous data ferries with partial observations. In: Proceedings of the eleventh ACM international symposium on mobile ad hoc networking and, computing, pp 141–150
18. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
19. Tisue S, Wilensky U (2004) Netlogo: a simple environment for modeling complexity. In: Proceedings of international conference on, complex systems, pp 16–21
20. Tseng Y-C, Ni S-Y, Chen Y-S, Sheu J-P (2002) The broadcast storm problem in a mobile ad hoc network. Wirel Netw 8(2/3):153–167
21. Wehner R, Srinivasan M (1981) Searching behaviour of desert ants, genuscataglyphis (formicidae, hymenoptera). J Comp Physiol 142(3):315–338
22. Pattem S, Krishnamachari B, Govindan R (2004) The impact of spatial correlation on routing with compression in wireless sensor networks. In: Information processing in sensor networks. Third international symposium on IPSN 2004, pp 28–35
23. Mller M, Wehner R (1994) The hidden spiral: systematic search and path integration in desert ants, cataglyphis fortis. J Comp Physiol A 175(5):525–530