Journal of Information Hiding and Multimedia Signal Processing ©2017 ISSN 2073-4212
Ubiquitous International Volume **8**, Number **1**, January **2017**

Fault Tolerant Scheduling Algorithm in Distributed Sensor Networks

Hongxuan Duan

Northeast Petroleum University
No. 500, Middle of Hebei Str., Haigang District, Qinhuangdao, China
duanhx@nepu.edu.cn

Yan Zhou

Henan Institute of Engineering
Zhengzhou Henan, 451191, China

Miaomiao Liu

Northeast Petroleum University
No. 500, Middle of Hebei Str., Haigang District, Qinhuangdao, China

Received March, 2016; revised June, 2016

Abstract. *Nowadays, distributed sensor networks have been deployed in many fields. Sensors are usually energy limited, and thus they must be scheduled effectively by making some of them sleep to keep the whole distributed sensor network work properly. At the same time, sensors are usually very cheap and they fail easily, but less working sensors (or more sleeping sensors) will make a network get wrong results. In this paper we studied how to schedule sensors effectively to make the whole distributed sensor network consume less energy while working fault tolerant at the same time. We designed a fault tolerant scheduling model for distributed sensor networks, and proposed a deferred active backup copy scheduling algorithm. We validated our proposed approach via massive simulation experiments.*

Keywords: Distributed sensor networks, Scheduling algorithm, Fault tolerant, Redundant backup

1. **Introduction.** Distributed sensor networks consist of massive small and energy limited sensors, where sensors communicate with each other with wireless communication protocols [1]. Sensors are placed manually or automatically by airplanes, and data are transmitted with multi-hop protocol among sensors and finally to the remote base station [2].

In distributed sensor networks, the power of sensors, or called nodes, is supplied with batteries, and thus is very limited, so how to maximize the life of distributed sensor networks by saving energy consumption of sensors is a key problem [3, 4, 5, 6]. The commonly used method to save energy consumption is scheduling working nodes by turns, and this method is reasonable. Sensors are placed very densely, and the data among sensors are redundant, so setting some sensors be idle don't affect the accuracy of data transmitted. However, sensors are made up of low price hardware, and these hardware go wrong frequently, so these sensors fail easily [7]. When a sensor fails, the data in it can be missing or dirty [8].

In order to make a distributed sensor network work properly, the working sensors need to be minimized, while at the same time keeping the data among them fault tolerant. In this paper, we assume each sensor as a distributed processor, and model the distributed sensor network with a fault tolerant scheduling model. For processing the backup copy tasks, we proposed a deferred active backup copy scheduling algorithm.

The rest of the paper is organized as follows. In section 2, we review related works about energy saving and fault tolerant scheduling algorithms in distributed sensor networks. In section 3, we propose a fault tolerant scheduling model in distributed sensor networks. In section 4, we propose a scheduling algorithm for working sensors. Experiments and conclusion are given in sections 5 and 6 respectively.

2. **Related works.**

In this section, we review related works about energy saving and fault tolerant scheduling algorithms in distributed sensor networks.

2.1. **Energy saving by scheduling working nodes.**

By scheduling working nodes, the energy of a distributed sensor network can be saved to maximize the life of the whole network. Currently, this kind of researches can be classified into certain sleeping and randomly sleeping. In both of the above two classifications, sleeping and working nodes work and sleep by turns to keep the whole network running properly.

In certain sleeping, some selected nodes sleep a certain time, and the other nodes work to keep the whole network working. In [9], Tian et al. let each node decide its status according to whether it can manage its neighbours. If one node can manage all of its neighbours, then it is a working node, otherwise, it sleeps. Ye et al. [10] propose a detection based network control protocol PEAS. Xu et al. [11] partition the whole network into virtual subnetworks, and keep one working node in each of these subnetworks. Zhang et al. [12] study how to coverage the whole network with less working nodes. In addition, Wang et al. [13] and Huang et al. [14] study the k-coverage problem of distributed sensor networks, and Jin et al. [15] study how to locate sensors of distributed sensor network before partitioning.

The basic idea of randomly sleeping is that, each node sleeps with the probability of p, and works with the probability of $1 - p$. The focus of this kind of study is the sleeping probability of nodes, node sensing radius and the region of subnetwork [16, 17]. In randomly sleeping, each node sleeps with fixed probability, so the whole network has lower adaptability.

2.2. **Fault tolerant scheduling in distributed systems.**

Classical fault tolerant techniques in distributed systems include distributed voting [18], rollback recovery [19] and backup [20]. However, these techniques don't take run-time into consideration. In order to satisfy run-time and fault-tolerant requirements, primary/backup copy is widely used in distributed system. According to storing and processing data in backup sensors, distributed sensor networks can finished specified work in run-time and fault tolerant. The primary/backup techniques can be classified into active backup copy [21], passive backup copy [22] and overlapping backup copy [23, 24].

The method of active backup copy can be easily implemented, and has no time requirement for the running time of tasks, but the deficiency is that it needs twice time of that under non-faulty situation [21]. In method of passive backup copy [21], the backup copy runs only when the primary copy fails. The advantage is that it doesn't run backup copy under non-faulty situation, but the disadvantage is that it needs synchronous overhead between primary and backup copies. The method of overlapping backup copy has both advantages of the above two methods. Karim et al. [23] propose a clustering based method for partitioning the whole network and choosing the primary copies. Geeta et al.

[24] apply an automatic network partitioning method for choosing primary and backup copites, and propose a dynamic method for adjusting the power consumption.

3. **Fault tolerant scheduling model.** In this section, we propose a fault tolerant scheduling model for distributed sensor networks.

Considering each sensor as a processor and each data processing as a task in that processor, then we have a group of tasks

$$\Gamma = \{\tau_1, \tau_2, \tau_3, \cdots, \tau_N\}, \tag{1}$$

$$\tau_i = (C_i, T_i), i = 1, 2, \cdots, N. \tag{2}$$

where N is the number of periodic tasks, C_i is the maximum running time of task τ_i, T_i is the period of task τ_i. The period of each task τ_i equals to its limit, and periods of different tasks are independent with each other. All tasks are preemptive, and the tasks of the same sensor is scheduled according to their priorities [25].

The backup copies of periodic tasks can be describe as follows.

$$B\Gamma = \{\beta_1, \beta_2, \beta_3, \cdots, \beta_N\}, \tag{3}$$

$$\beta_i = (D_i, T_i), i = 1, 2, \cdots, N. \tag{4}$$

For each task τ_i, these is a backup copy β_i, the period of β_i is the same as its primary copy task. As β_i is the backup of τ_i, we have that $D_i \leq C_i$. The primary and backup copies of a task are scheduled to different sensors. In the following, we denote γ_i as either the primary copy or the backup copy, i.e. $\gamma_i = \tau_i$ or $\gamma_i = \beta_i$.

There are three running models for backup copies of tasks in scheduling models, and they are active backup copy, deferred active backup copy and passive backup copy. Let $Status(\beta_i)$ denotes the running model of the backup copy β_i, then we have $Status(\beta_i) \in \{$ common-active, passive, deferred-active $\}$. In this paper, we applied active backup copy and passive backup copy described by Ghosh et al. [26], and propose a deferred active backup copy method described in the next section.

The set of sensors or processors is

$$P = \{P_1, P_2, P_3, \cdots, P_M\}, \tag{5}$$

where P_i is the i-th sensor, and M is the total number of sensors in a distributed sensor network. In this paper, we assume that all sensors are the same, and each task has the same running time on different sensors. Let $P(\tau_i, \beta_i)$ denotes the sensor that runs τ_i or β_i. While detecting faults of sensors, we apply the acceptance test described in [27].

4. **Deferred active backup copy scheduling algorithm.** In this section, we propose a deferred active backup copy scheduling algorithm. The idea of the proposed algorithm is similar to [22], but the difference is that we defer the backup copy to the end, so some backup copies can be run passively.

Figure 1 is an illustration of the proposed deferred active backup copy. As can be seen from the figure that, the worst case running time of τ_i is R_i, and the worst case running time of β_i is BR_i. During each period $[hT_i, (h+1)T_i]$ of τ_i, we divide its running time into the redundant part Rp_i and the backup part Bp_i. Rp_i and the primary copy run in parallel, and Bp_i runs only when the primary copy fails. However, if the sensor of the primary copy fails, the recovery time B_{ij} ($B_{ij} = T_i - R_{ij}$) on backup sensor cannot be left for β_i, because it can be interrupted by other tasks with high priorities. So, we must compute how much of B_{ij} can be left to β_i. If it is bigger than 0, then leave it for the backup of β_i, i,e, $BackT(i)$. When all sensors work properly, the redundant part of β_i runs in parallel with the primary copy, and then the deferred active backup copy

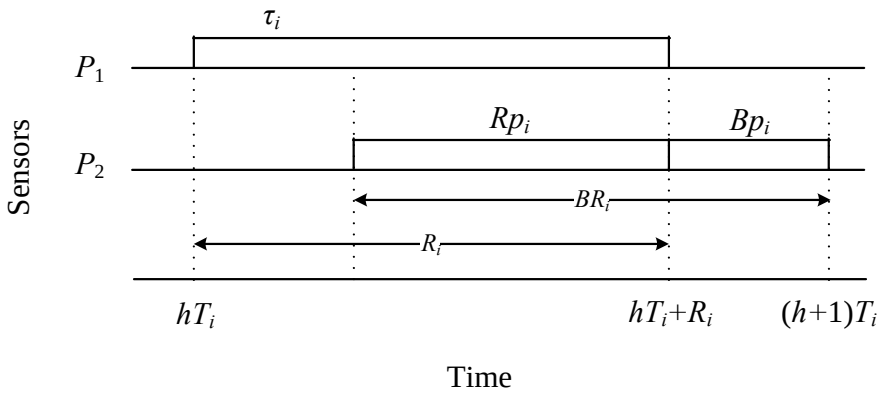

FIGURE 1. Illustration of deferred active backup copy

can be considered as a period task with period T_i, ending time R_{ij} and running time $D_i - BackT(i)$.

4.1. **Computing running time of backup copy.** Let $Primary(P_j)$ and $Backup(P_j)$ denote sets of primary and backup copy tasks scheduled to P_j respectively, and $deferred\_active(P_j)$ and $common\_active(P_j)$ denote the deferred active and common active tasks scheduled to P_j respectively.

If the passive backup copy is scheduled to P_j and the primary copy is scheduled to P_f, then we represent these tasks as the set $passiveRecover(P_j, P_f)$; if the common active copy is scheduled to P_j, and the primary copy is scheduled to P_f, then we represent these tasks as the set $common\_activeRecover(P_j, P_f)$; and if the deferred active copy is scheduled to P_j, and the primary copy is scheduled to P_f, then we represent these tasks as the set $deferred\_activeRecover(P_j, P_f)$. In addition, we have

$$\begin{aligned} Recover(P_j, P_f) = {} & passiveRecover(P_j, P_f) \cup \\ & common\_activeRecover(P_j, P_f) \cup \\ & deferred\_activeRecover(P_j, P_f). \end{aligned} \tag{6}$$

In a deferred active backup copy, whether or not the active backup copy runs actively is the key problem. Because the tasks are scheduled according to their priorities, running time of backup copy is the idle time of primary copy while recovering. This idle time $ComputeFreeTime(B_{ij}, k)$ can be computed in the following three steps.

1. If the sensor P_j which runs task τ_i fails, compute the task set of $\sigma = Primary(P_k) \cup Recover(P_k, P_j)$ on P_k;
2. sort the tasks in σ by the descending order, compute the time of each task in σ between $[B_{ij}, T_i]$ according to priorities of tasks, and get the total time $TimeOccupy$ of σ in $[B_{ij}, T_i]$;
3. compute the idle time that is left for β_i, i.e. $BackT(i) = B_{ij} - TimeOccupy$.

While β_i is scheduled to sensor P_k, if sensor P_j, which runs task τ_i, fails, then we can get the idle time that is left to β_i on sensor P_k in B_{ij}.

4.2. **Deciding type of backup copy.** While scheduling or allocating each task, we first schedule its primary copy, and then schedule its backup copy. The scheduling of the backup copy can be described as the following equation. $Stasus(\beta_i) =$

$$\begin{cases} passive, B_{ij} \geq D_i, \\ common\_active, B_{ij} < D_i, BackT(i-1) = 0, \\ deferred\_active, B_{ij} < D_i, BackT(i-1) > 0, \end{cases} \tag{7}$$

where $P(\tau_i) = P_j$, $P(\beta_i) = P_k$ and $BackT(i-1) = ComputeFreeTime(B_{ij}, k)$.

When running time of β_i is equal to or smaller than B_{ij}, the backup copy is passive; and when running time of β_i is bigger than B_{ij}, $Status(\beta_i)$ is determined by $BackT(i-1)$ in B_{ij}. If $BackT(i-1) = 0$, β_i is an active backup copy, and if $BackT(i-1) > 0$, then some backup copies are postponed and β_i is a deferred active backup copy.

4.3. **Scheduling periodic tasks.** In this paper, we schedule the primary and backup copies of tasks with "best adaptability" and "first adaptability" strategy. The main idea is reducing the worst case response time while processing the primary copy, and thus reducing unnecessary redundancy of backup copies under non-faulty situation by information of primary copy earlier.

The strategy of task scheduling is that, sort primary and backup copies of tasks according to ascending order of periods first, and let the priorities of the primary copies be bigger than the backup copies. The order after sorting is

$$\tau_1, \beta_1, \tau_2, \beta_2, ...\tau_N, \beta_N. \tag{8}$$

According to the above order (from τ_1 to β_N), we schedule the primary copy τ_i, and then the backup copy β_i for each task i. For the primary copy, we apply the "best adaptability" strategy, that is scheduling τ_i to M sensors, computing the worst case response time R_{ij} of task τ_i on sensor P_j. Assuming that the minimum response time of task τ_i on sensor P_k $(1 \leq k \leq M)$ is R_{ik}, if $R_{ik} > T_i$, then τ_i cannot be scheduled to the above M sensors, so we startup a new sensor to schedule τ_i, and let $M = M + 1$ and $P(\tau_i) = P_M$; and if $R_{ik} < T_i$, then we judge the type of backup copy via equation (4), and let $P(\tau_i) = P_k$. After that, we find sensors that are suitable for scheduling β_i from the first sensor P_m ($m = 1, 2, ..., M, m \neq k$). If the above sensors exist, then let $P(\beta_i) = P_m \neq P(\tau_i)$, otherwise, we startup a new sensor and let $M = M + 1$ and $P(\beta_i) = P_M$.

As we sort tasks by the descending order of priorities, we only need to check the schedulability of tasks at each task allocation. At the same time, we allocate tasks under the no sensor failure and one failure of any sensor two situations.

5. **Simulation experiments.** Status of sensors affects the energy consumption, and also reflects whether or not sensors are faulty. More sleeping sensors mean less energy consumption of the whole network, but also less resistant ability to sensor faults. In this experiments, we mainly focus on comparison of sleeping rate and faulty rate of sensors.

The simulation experiments are implemented in a 300×300 unit region, where sensors are placed randomly with radius 40 unit. We use histograms to illustrate the sleeping rate (SR), and curve figures to illustrate coverage rate (CR). Here, the less CR is, the more likely sensors fails.

Figure 2 and 3 illustrate sleeping rate and coverage rate of sensors of our method under different probabilities (p) of sensor sleeping. As can be seen from the figures that, however p is, SR is nearly the same as p, so when p increases, the number of working sensor increases, and this would make more sensors be redundant. For example, when the number of sensors increases from 200 to 300, if $p = 50\%$, then there are 100 and 150 sensors working.

In addition, we implement three scheduling strategies in equation 8, and the results are in figures 4-7, 8-9 and 10-11 respectively. Figures 4, 5, 6 and 7 are the DR and CR of the

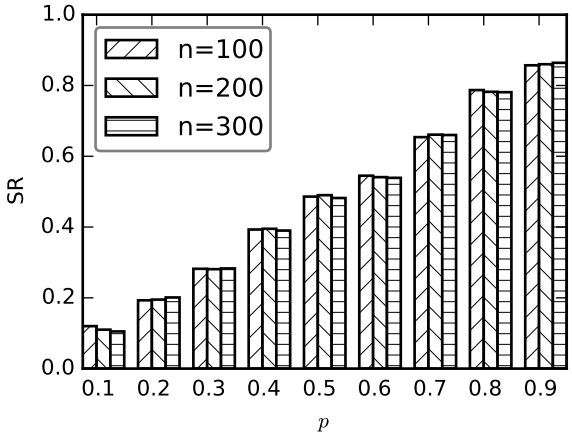

FIGURE 2. Constant probability scheduling (SR)

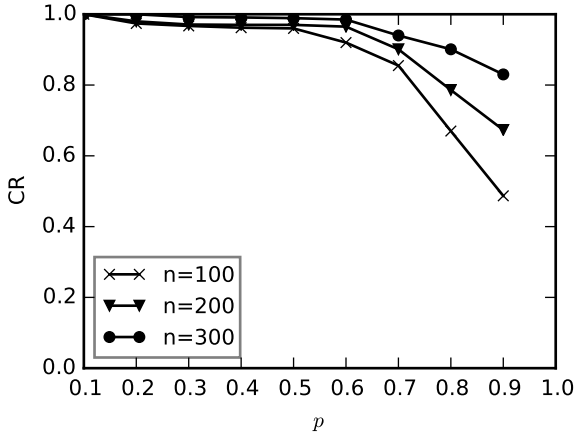

FIGURE 3. Constant probability scheduling (CR)

passive backup copy method under different k and sensor number. When we increase the backup number k (in figures 4 and 5), the coverage rate of the whole network increases too, but the sleeping rate decreases. When the density of the network increases (in figures 6 and 7), the sleeping rate increases too, and the reason of this phenomenon is the same as the constant probability scheduling method. We let k be fixed, and observe how coverage is affected by node number (density of the network). When k is small, the sleeping rate is very high, but the coverage rate is low. The reason is that the value of k is unreasonable, and neighbourhood of sensors cannot be updated in real time.

Figures 8 and 9 are results of the common active recover method. In this experiment, we supervise scheduling with dynamic sensor information. In figure 8, the sleeping rate increases with the number of sensors. In figure 9, different settings of k have little effect on the network coverage, and the only exception is $k = 1$, where there is no backup copy.

Figures 10 and 11 are results of the deferred active recover method. In this experiment, we supervise scheduling of sensors with distances to its neighbours. Figure 10 illustrates the change of sleeping rate of sensors along with the number of nodes, and figure 11 is the coverage rate of sensor with respect to the number of nodes. As can be seen from the two figures that, the sleeping rate of sensors increases with the node number, and the coverage rate stabilizes when the node number reaches 150. This reflects that the node number has little effect on the coverage rate in a stable sensor network.

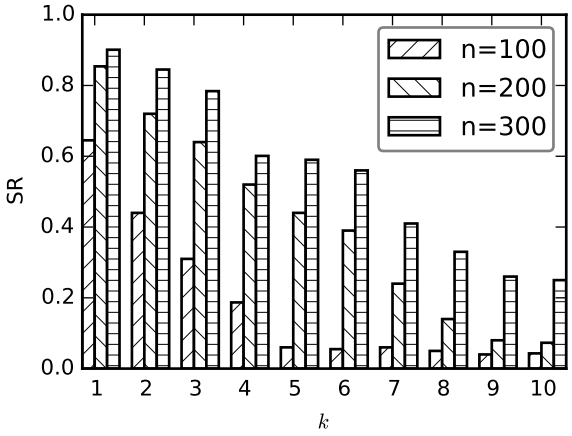

FIGURE 4. Passive scheduling performance based on backup number (SR)

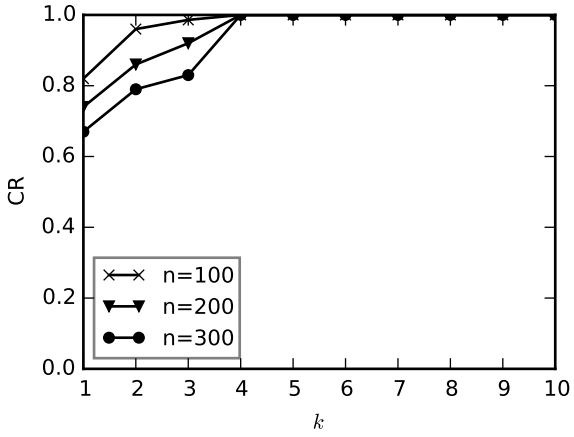

FIGURE 5. Passive scheduling performance based on backup number (CR)

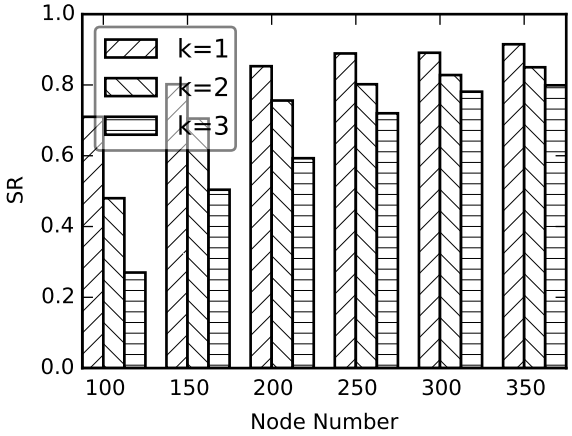

FIGURE 6. Passive scheduling performance based on neighbour number (SR)

At last, we compare our method with the DT (Dynamic Time) [**?**] method scheduling method. The DT method clusters sensors into groups according to their locations, and use sensors of the same cluster to backup. Figure 12 illustrates the sleeping rate comparison of the two methods. In our model, we let $\sigma = 0.2$ and $k = 2$ respectively. From this figure

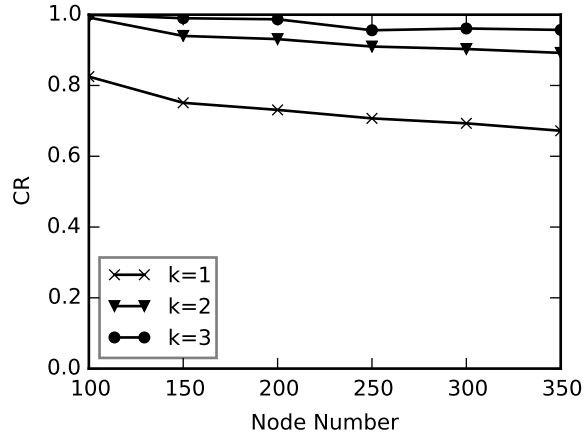

FIGURE 7. Passive scheduling performance based on neighbour number (SR)

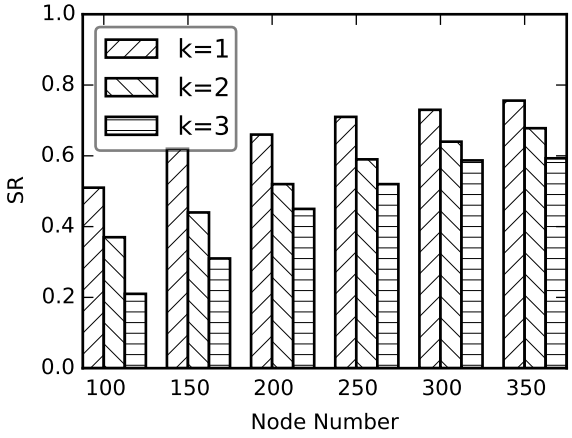

FIGURE 8. Common active scheduling performance based on node number (SR)

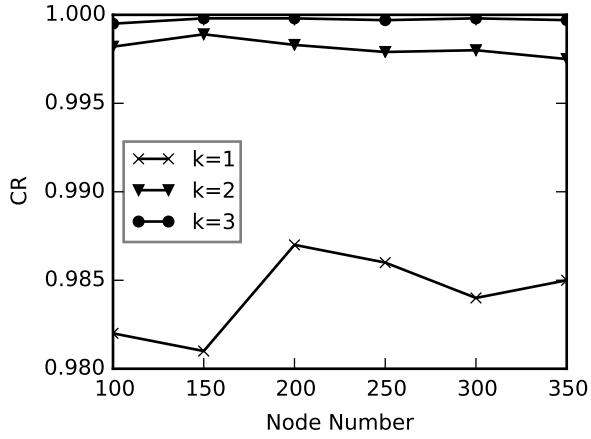

FIGURE 9. Common active scheduling performance based on node number (CR)

we can see that, $\sigma = 0.2$ and $k = 2$ have similar SR, and both of them are bigger than the DT method.

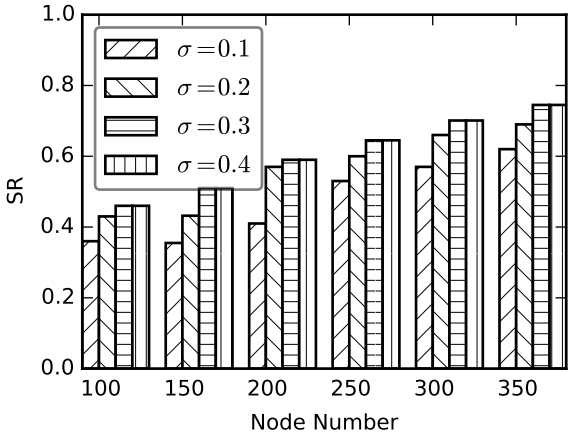

FIGURE 10. Deferred active scheduling performance based on neighbour distance (SR)

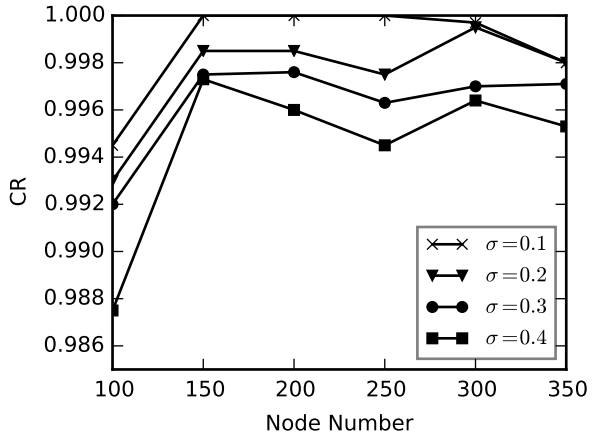

FIGURE 11. Deferred active scheduling performance based on neighbour distance (CR)

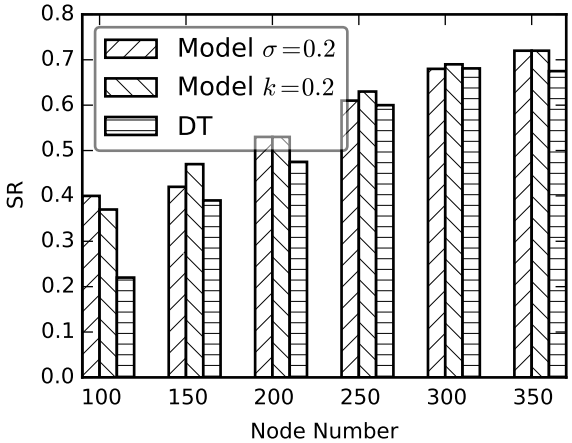

FIGURE 12. Ours vs. position based

6. **Conclusion.** Distributed sensor networks are ubiquitous and applied in many fields. In order to handle the contradiction of less working sensors and more easily faults, we designed a fault tolerant scheduling model for distributed sensor networks, and proposed

a deferred active backup copy scheduling algorithm. Massive simulation experiments validated the effectiveness of our proposed approach.

In the future, we will deploy a distributed sensor network in a real scenario, and validate the efficiency and real-time ability of the proposed algorithm with real applications.

Competing interests. The authors declare that they have no competing interests.

Acknowledgement. This work was supported by Qinhuangdao Science and Technology Bureau Supporting Project (No. 201502A003).

REFERENCES

[1] V. Lesser, C. L. Ortiz Jr and M. Tambe. Distributed sensor networks: A multiagent perspective. *Springer Science & Business Media,* vol. 9, pp. 132-153, 2012.

[2] R. Sumathi and M. Srinivas, A survey of QoS based routing protocols for wireless sensor networks, *Journal of Information Processing Systems*, vol. 8, no. 4, pp. 589-602, 2012.

[3] A. A. Aziz, Y. A. Sekercioglu, P. Fitzpatrick and M. Ivanovich, A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks, *Communications Surveys & Tutorials, IEEE,* vol. 15, no. 1, pp. 121-144, 2013.

[4] H.-C. Wu, S.-C. Huang and C.-Y. Lin, MEMS Sensors Applied in Finswimming Movement Analysis. Journal of Computers and Applied Science Education, vol. 2, no. 1, pp. 32-44, 2015.

[5] T.-T. Nguyen, T.-K. Dao, M.-F. Horng and C.-S. Shieh, An Energy-based Cluster Head Selection Algorithm to Support Long-lifetime in Wireless Sensor Networks, *Journal of Network Intelligence*, vol. 1, no. 1, pp. 23-37, 2016.

[6] F.-C. Chang and H.-C. Huang, A Survey on Intelligent Sensor Network and Its Applications. Journal of Network Intelligence, vol. 1, no. 1, pp. 1-15, 2016.

[7] E. Ould-Ahmed-Vall, B. H. Ferri and G. F. Riley, Distributed fault-tolerance for event detection using heterogeneous wireless sensor networks, *Mobile Computing, IEEE Transactions on,* vol. 11, no. 12, pp. 1994-2007, 2012.

[8] C.-H. Yang, G. Deconinck and W.-H. Gui. Fault-tolerant scheduling for real-time embedded control systems, *Journal of Computer Science and Technology*, vol. 19, no. 2, pp. 191-202, 2004.

[9] D. Tian and N. D. Georganas, A coverage-preserving node scheduling scheme for large wireless sensor networks. Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, ACM, pp. 32-41, 2002.

[10] F. Ye, G. Zhong, J. Cheng, S. Lu and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks, *Distributed computing systems, 2003. Proceedings. 23rd international conference on. IEEE*, pp. 28-37, 2003.

[11] Y. Xu, J. Heidemann and D. Estrin, Geography-informed energy conservation for ad hoc routing. *Proceedings of the 7th annual international conference on Mobile computing and networking. ACM,* pp. 70-84, 2001.

[12] H. Zhang and J. C. Hou, Maintaining sensing coverage and connectivity in large sensor networks, *Ad Hoc & Sensor Wireless Networks,* vol. 1, no. 1-2, pp. 89-124, 2005.

[13] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless and C. Gill, Integrated coverage and connectivity configuration in wireless sensor networks, *Proceedings of the 1st international conference on Embedded networked sensor systems. ACM*, pp. 28-39, 2003.

[14] C.-F. Huang and Y.-C. Tseng, The coverage problem in a wireless sensor network. *Mobile Networks and Applications,* vol. 10, no. 4, pp. 519-528, 2005.

[15] Z. Jin, D. Shi, Q. Wu and H. Yan, Random Walk Based Location Prediction in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks,* vol. 2013, 2013.

[16] S. Kumar, T. H. Lai and J. Balogh, On k-coverage in a mostly sleeping sensor network, *Proceedings of the 10th annual international conference on Mobile computing and networking. ACM,* pp. 144-158, 2004.

[17] S. Shakkottai, R. Srikant and N. B. Shroff, Unreliable sensor grids: Coverage, connectivity and diameter, *Ad Hoc Networks,* vol. 3, no. 6, pp. 702-716, 2005.

[18] L. Xu and J. Bruck, Deterministic voting in distributed systems using error-correcting codes, *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 8, pp. 813-824, 1998.

[19] T.-H. Lin and K. G. Shin. Damage assessment for optimal rollback recovery, *IEEE Transactions on Computers*, vol. 47, no. 5, pp. 603-613, 1998.

[20] R. Davoli, L.-A. Giachini,O. Babaoglu, A. Amoroso and L. Alvisi, Parallel computing in networks of workstations with Paralex, *IEEE Transactions on Parallel and Distributed Systems,* vol. 7, no. 4, pp. 371-384, 1996.

[21] C. Yang and G. Deconinck, A fault-tolerant reservation-based strategy for scheduling aperiodic tasks in multiprocessor systems. *Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on. IEEE*, pp. 319-326, 2002.

[22] A. Bari, A. Jaekel, J. Jiang and Y. Xu, Design of fault tolerant wireless sensor networks satisfying survivability and lifetime requirements, *Computer Communications,* vol. 35, no. 3, pp. 320-333, 2012.

[23] L. Karim, N. Nasser and T. Sheltami, A fault-tolerant energy-efficient clustering protocol of a wireless sensor network, *Wireless Communications and Mobile Computing,* vol. 14, no. 2, pp. 175-185, 2014.

[24] D. Geeta, N. Nalini and R. C. Biradar, Fault tolerance in wireless sensor network using hand-off and dynamic power adjustment approach. *Journal of Network and Computer Applications,* vol. 36, no. 4, pp. 1174-1185, 2013.

[25] A. Li and N. Xie, A robust scheduling for reconfigurable manufacturing system using Petri nets and genetic algorithm, *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on. IEEE,* vol. 2, pp. 7302-7306, 2006.

[26] S. Ghosh, R. Melhem, D. Moss and J. S. Sarma, Fault-tolerant rate-monotonic scheduling, *Real-Time Systems,* vol. 15, no. 2, pp. 149-181, 1998.

[27] M. Goljan, J. Fridrich and T. Filler, Large scale test of sensor fingerprint camera identification. *IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics*, pp. 72540I-72540I, 2009.