Reliability Analysis of Software Defined Wireless Sensor Networks

Na Gong and Xin Huang(✉)

International Business School in Suzhou, Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, China
Na.Gong12@student.xjtlu.edu.cn, Xin.Huang@xjtlu.edu.cn

Abstract. Software defined wireless sensor networks (SDWSN), which separate the control and data forwarding functions, have been proposed to solve the management dilemma of conventional wireless sensor networks (WSN). This technique can achieve a dynamic and centralized network management. However, the study on its reliability is still deficient. In this paper, SDWSN is formally modeled using Continuous-Time Markov Chains (CTMCs), and its reliability is verified using probabilistic model checking techniques. The verification results are expected to give suggestions to the future SDWSN designs.

Keywords: Model checking · PRISM · Software defined network · Wireless sensor network

1 Introduction

Nowadays, as an important technique in the next generation wireless and mobile networks, the wireless sensor network (WSN) is becoming more and more complex and demanding [6,7,17]. It has been widely used in many areas, for example military, navigation and environmental monitoring [15]. Typical WSNs are self-configuring networks [10,15], and their topologies are frequently changed [11]. This feature of WSNs poses the demand for dynamic network management. Therefore, software-defined wireless sensor network (SDWSN), which is a combination of WSN and software-defined networking (SDN), is proposed [5,10,11,15,18,23].

SDN is a new paradigm of networking system that provides dynamic network management and configuration. Why SDN is needed? Traditional networks consist of many devices for forwarding and processing data packets [21]. It is extremely time-consuming to deploy and maintain traditional networks, because the packet routing control functions are located in routers, and they need to be updated locally in most cases [11]. To remedy this limitation, SDN is proposed. It separates the control plane and the data plane. The data plane composed by physical network devices is responsible to perform packet forwarding and processing functions. The control layer is in charge of the whole network management [22]. The applications hosted in the application layer can program the devices in data plane [14].

© Springer Science+Business Media Singapore 2016
S.Y. Ohn and S.D. Chi (Eds.): AsiaSim 2015, CCIS 603, pp. 65–78, 2016.
DOI: 10.1007/978-981-10-2158-9_6

Similar to SDN, SDWSN aims at separating the control plane from the data forwarding plane in WSNs. These days, the dramatically raising application of the wireless and sensing technology expands the requirement for the dynamic, flexible and secure management of WSN. Hence, to satisfy these needs, the Software-Defined Wireless Sensor Network (SDWSN) is proposed [5,6,11,15,18,23]. Compared with the traditional WSN, SDWSN has several advantages:

- The network becomes programmable. This allows for more dynamic network segmentation and utilization without changes in other devices [12].
- The centralized controller maintains a global view of the whole network. Thus, security and flexibility of SDWSN could be improved.

However, one major concern about SDWSN is its reliability. Because SDWSM has a central controller, it may becomes the bottleneck of the whole system. If it fails, the whole system will be down. However, so far, few researchers study the reliability of the SDWSN which however is not a negligible issue. This paper therefore aims to study the reliability of the SDWSN by using the probabilistic model checking technique with the tool PRISM.

The reminder of this paper is organized as follows. In Sect. 2, SDWSN is described. The probabilistic model checking technique is explained in Sect. 3. Section 4 describes the modeling of the SDWSN structure. Section 5 analyzes the results. Finally, Sect. 6 provides the conclusion.

2 Software Defined Wireless Sensor Network

The widespread application of WSN attracts much awareness from both the industrial and academic fields. Recently, many researchers focuses on enhancing the capability of WSN reconfiguration adaption. First of all, many MAC protocols such as T-MAC, C-MAC and snapMAC have been proposed [3,19,20]. Also, a centralized architecture is proposed for WSNs [9]. In addition, using SDN techniques in WSNs gains attentions of many researchers [5,6,10,11,15,18,23].

A SDN is composed by the following layers (shown in Fig. 1):

- Data plane: it consists of network devices such as switches to maintain the packet forwarding functionality [22].
- Controller plane: it is in charge of the whole network management especial the management of the packet processing rules [21].
- Application plane: it contains various applications such as Could Computing Service and Data Warehouse [1].
- APIs: it allows the centralized controller entity in the intermediate plane to transfer instructions and information from or to the data and application plane [4]. The OpenFlow protocol now is a popular one [12].

The construction of SDWSN is inspired by three important ideas in SDN, namely, the separation of controller and data planes, standardized application interfaces and centralized management. As presented in Fig. 2, the differences between our SDWSN and SDN are:

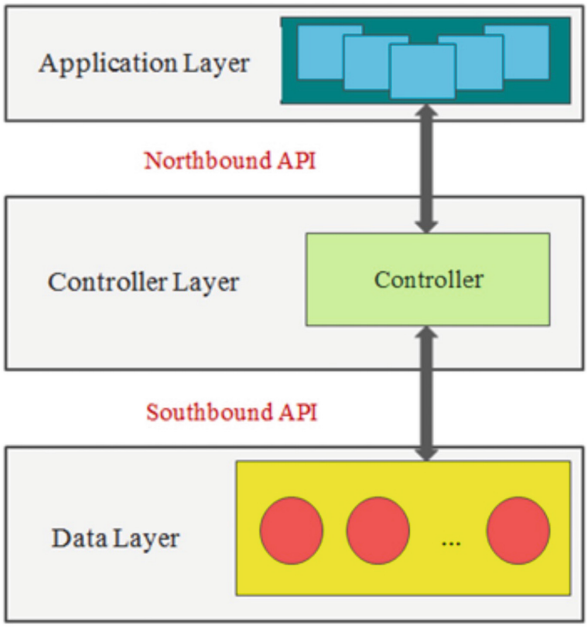

Fig. 1. The architecture of SDN

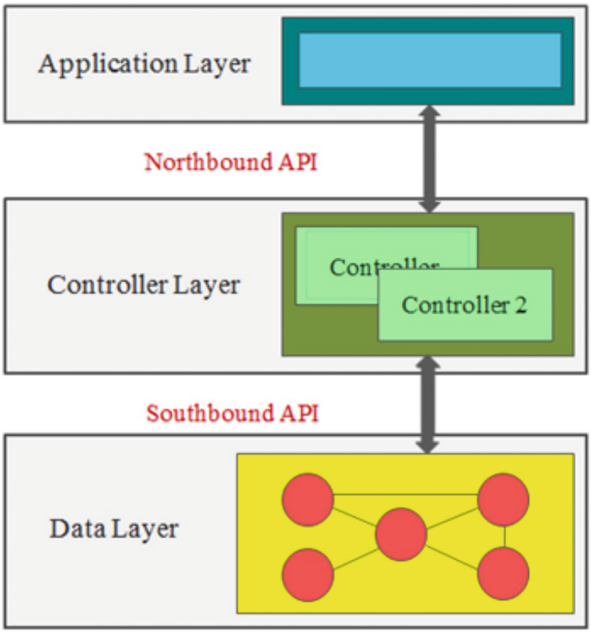

Fig. 2. The architecture of SDWSN

– Replaces the data plane in SDN with a WSN;
– Increases the number of controllers to enhance its reliability.

3 Probabilistic Model Checking

The behaviors of many real-life systems have stochastic characteristic. To model these systems, a formal verification technique, Probabilistic Model Checking, is widely used [13]. It is able to evaluate the correctness, performance and dependability of those systems [13]. Three procedures are involved in the model checking:

– Construction of a mathematical model.
– Specify properties of the system.
– Analysis these properties.

In the second step, the property specification is usually represented using probabilistic temporal logics. It usually uses specific operators to reason about the probability or Boolean value of events occurrences.

3.1 Continuous-Time Markov Chain

Continuous-time Markov Chains (CTMCs) usually describe a state transition system augmented with probabilities. Figure 3 is a typical dense model of time with discrete states. Its transitions among states could occur at any time unit in CTMC. These arrows present the state transferring with particular possibility in the system. The most important feature of this model is the Markov property that is "memorylessness". This could be described as the following equation:

$$\mathrm{P}(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \mathrm{P}(X_{n+1} | X_n = x_n)$$

where X_n stands for the state of the system at time n and $\mathrm{P}(X_n = x_n)$ is the corresponding probability. The probability distribution of system state at time $n+1$ (the future state) depends only on the state at time n (the current state) and does not relate with the states at time $(1, n)$ (the past states).

A continuous-time Markov chain (CTMC) M is a four-tuple (S, s, R, L). Assume each component of the system has n states, then S is a finite set of states where $S = \{s_0, s_1, s_2, \ldots, s_n\}$ and $s \in S$; s is the initial status of the component; R is the transition rate matrix of $|S| \times |S|$, and L is the labeling function with atomic propositions [2]. Figure 3 is an example of CTMC:

$$S = \{s_0, s_1, s_2, s_3, s_4\};$$

$$s = s_0;$$

$$R = \begin{bmatrix} 0 & 0.5 & 0.2 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0.2 \end{bmatrix};$$

$$L(s_0) = \{\text{OK}\}, L(s_1) = L(s_3) = L(s_4) = \{\}, L(s_2) = \{\text{Failed}\}, L(s_5) = \{\text{Waiting}\}.$$

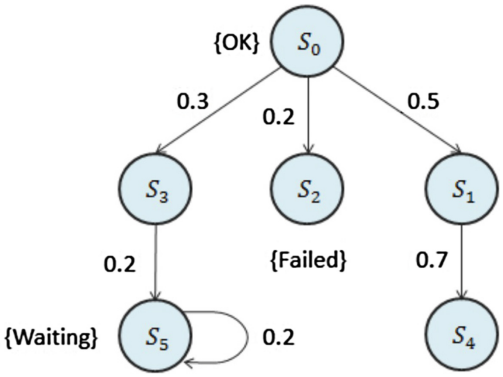

Fig. 3. CTMC example.

Time in CTMC is generally modeled using exponential distribution [16]. For a continuous random variable X, if its density function is given by:

$$f(t) = \begin{cases} \lambda e^{-\lambda t}, \text{ if } t > 0 \\ 0, \quad \text{otherwise} \end{cases}$$

where λ is the rate, X is exponential with the parameter $\lambda > 0$. In this system, all delays and repair time are assumed as X here, therefore, the property $\mathrm{P}(X \leq t)$ can be computed by

$$F(t) = \mathrm{P}(X \leq t) = \int_0^t \lambda e^{-\lambda t} \mathrm{d}x$$

and the property $\mathrm{P}(X > t)$ is $e^{-\lambda t}$.

3.2 Continuous Stochastic Logic

Continuous stochastic logic (CSL) is used for specifying properties of CTMCs. It is defined by two types of syntaxes: state formula (Φ) and path formula (Ψ). CSL allows many reliability related properties to be specified such as the reachability property [13]. This usually outputs a failure rate where the transaction of the system modeled by CTMC finally shut down, or a success rate. In this project, we mainly focus on the failure rate of the system.

3.3 PRISM

PRISM is a popular probabilistic model checking tool. It now mainly supports four types of probabilistic model: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), Markov decision processes (MDPs) and Probabilistic timed automata (PTAs) [8].

The model in PRISM is specified using the PRISM modeling language. Each System is described as modules. Each module contains its local variables which is used to indicate the possible states. The status transition behavior in modules is represented in command. For example:

```
[] x=0 -> 0.8:(x'=0) + 0.2:(x'=1);
```

This command claims that when the variable x has value 0, the update state will remain at 0 with probability 0.8 and change into 1 with probability 0.2. Additionally, the module uses cost and reward to permit reasoning about the quantitative measure of the system such as the expected power consumption and expected time.

PRISM's property specification language is based on temporal logics, namely, PCTL, CSL, probabilistic LTL and PCTL* [13]. It mainly has three operators:

- `P` operator: this is used to reason about the probability of an event occurrence, e.g. what is the probability of $y = 6$:
  `P=? [y = 6].`
- `S` operator: this refers to the steady-state behavior of the model, e.g. what is the long-run probability of the queue being more than 75 % full:
  `S=? [q > 0.75].`
- `R` operator: this is used for reasoning about the expected cost of the model. For example: what is the expected number of lost requests within 15.5 time units
  `R=? [C <= 15.5].`

4 Model Checking SDWSN

This section illustrates a SDWSN formal model and presents its implementation details.

4.1 Formal Model of a Typical SDWSN

PRISM is used to analyze SDWSN reliability. To achieve this, the general SDN architecture shown in the Fig. 1 is modified into the structure presented in the Fig. 2. The improved SDWSN has following features:

- Application Layer: Because this paper does not consider the effects of applications, all of them are regarded as a whole with very low failure rate.
- Controller Layer: If the controller fails, the system will shut down. Hence, the modified structure adds one controller to increase the reliability.
- WSN Layer: This consists of several sensors to collect and measure information and other physical quantities. Similar with control plane, if all sensors fail, then controller will shut down the system.

4.2 Implementation

In the formal model, two modules (controller layer and WSN layer) are realized. The detail structures of modules are as follows:

- Module 1 - Controller layer: Two controllers are c_1 and c_2. The states are: both two controllers are working ($m = 2$), only one is working ($m = 1$) and fail ($m = 0$). Each controller has the same failure rate λ_c. The transition rate of the controller layer whose initial status is $m = 2$ to the state $m-1$ is $m \times \lambda_c$.
- Module 2 - WSN layer: At least two sensors are considered in this module. Additionally, the number of sensors can be manually changed. Each sensor has the same failure probability λ_a.

To verify the dependability of this model, the following failure types are considered:

- Controller layer fails if none of controllers is working.
- WSN fails if none of sensors is working.
- System fails if the controller layer fails.
- System fails if the number of working sensors is less than the lower bound.
- System fails if the skipping cycle is greater than the upper limit. (The controller can automatically skip the current operation cycle when the application or WSN layer is abnormal, then a `Max_COUNT` is set to limit the skipping.)

5 Results

The main results are presented in this section. The factors of influencing the reliability of SDWSN are also analyzed.

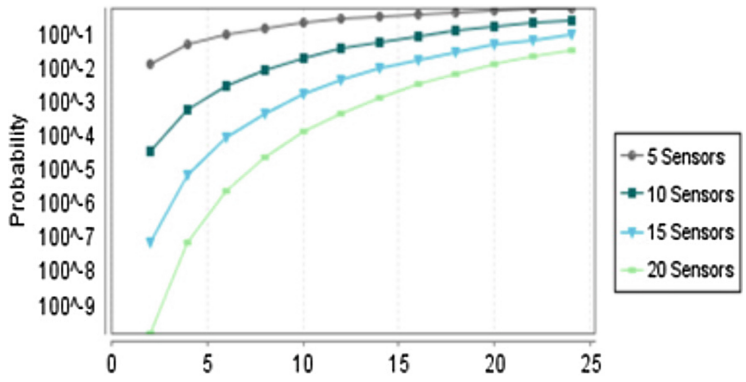

Fig. 4. The performance of WSN layer in 24 h with different number of sensor nodes. The y-axis is the failure probability of the WSN layer; and the x-axis is the duration (24 h). The number of sensor nodes is represented in the right; e.g., "5 sensors" means that the WSN layer contains 5 sensors. The failure rate of each sensor node is once a day.

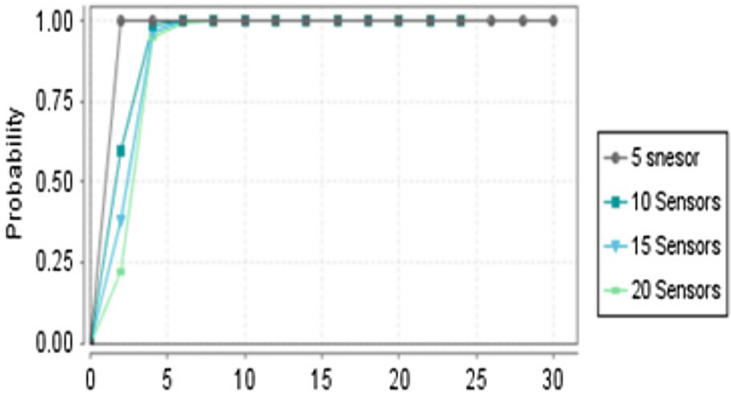

Fig. 5. The performance of WSN layer in 30 days with different number of sensor nodes. The y-axis is the failure probability of the WSN layer; and the x-axis is the duration (30 days). The number of sensor nodes is represented in the right; e.g., "5 sensors" means that the WSN layer contains 5 sensors. The failure rate of each sensor node is once a day.

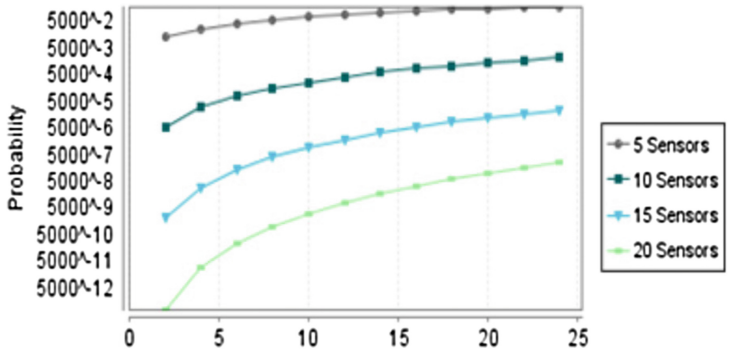

Fig. 6. The performance of WSN layer in 24 h with different number of sensor nodes. The y-axis is the failure probability of the WSN layer; and the x-axis is the duration (24 h). The number of sensor nodes is represented in the right; e.g., "5 sensors" means that the WSN layer contains 5 sensors. The failure rate of each sensor node is once per 30 days.

In Fig. 4, the failure rate of the WSN layer is shown. Its setting is:

- Failure rate of each sensor: once per day;
- Duration: 24 h;
- The number of sensors: 5, 10, 15, and 20.

As shown in Fig. 4, increasing the number of sensors can decrease the failure probability of the entire WSN layer. However, if we set the duration to 30 days, this strategy will be challenged. The result is shown in Fig. 5. From this figure, we can see that the failure probability of the WSN plane is 100 % after 4 days.

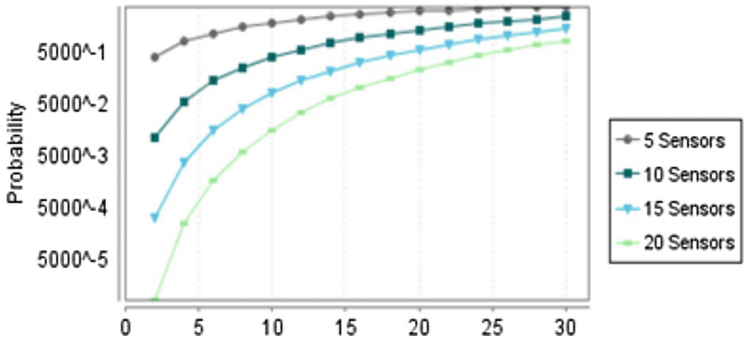

Fig. 7. The performance of WSN layer in 30 days with different number of sensor nodes. The y-axis is the failure probability of the WSN layer; and the x-axis is the duration (30 days). The number of sensor nodes is represented in the right; e.g., "5 sensors" means that the WSN layer contains 5 sensors. The failure rate of each sensor node is once per 30 days.

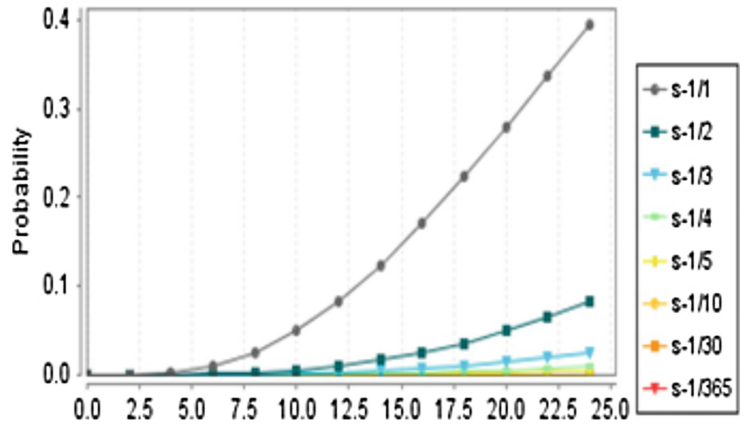

Fig. 8. The performance of WSN layer in 24 h with different failure rate of individual sensor nodes. The y-axis is the failure probability of the WSN layer; and the x-axis is the duration (24 h). The failure rate of sensor nodes is represented in the right; e.g., "s-1/2" means that the failure rate is once per 2 days. The number of sensor nodes is 5.

This suggests that when the failure probability of individual sensor node is high (once per day), increasing the number sensor nodes is not a good way of enhancing the reliability of the WSN layer.

In Figs. 6 and 7, the failure rate of each sensor in the WSN plane is changed to once per month. Compared with Figs. 4 and 5, the failure probability of the WSN plane is significantly lower. This depicts that decreasing the failure rate of each sensor node can decrease the failure probability of the sensor layer.

In order to study the relation between the sensor node failure rate and the WSN layer failure rate, we did several new experiments. In Figs. 8 and 9, we can

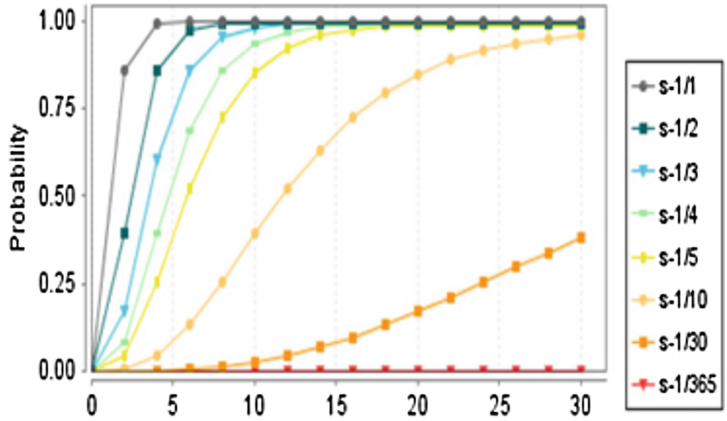

Fig. 9. The performance of WSN layer in 30 days with different failure rate of individual sensor nodes. The y-axis is the failure probability of the WSN layer; and the x-axis is the duration (30 days). The failure rate of sensor nodes is represented in the right; e.g., "s-1/2" means that the failure rate is once per 2 days. The number of sensor nodes is 5.

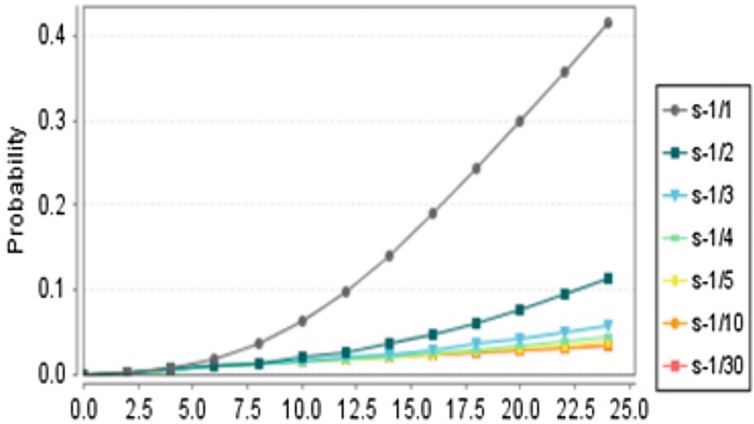

Fig. 10. The performance of the whole system (SDWSN) in 24 h with different failure rate of individual sensor nodes. The y-axis is the failure probability of the system; and the x-axis is the duration (24 h). The failure rate of sensor nodes is represented in the right; e.g., "s-1/2" means that the failure rate is once per 2 days. The number of sensor nodes is 5. The failure rate of the controller is?

see that lower sensor node failure rate is corresponding to lower WSN failure rate. In Figs. 10 and 11, we can see that lower sensor node failure rate can also enhance the system (SDWSN) reliability given a fixed controller failure rate.

Except the sensor nodes, the system reliability is also related to controllers' status. We have the following findings:

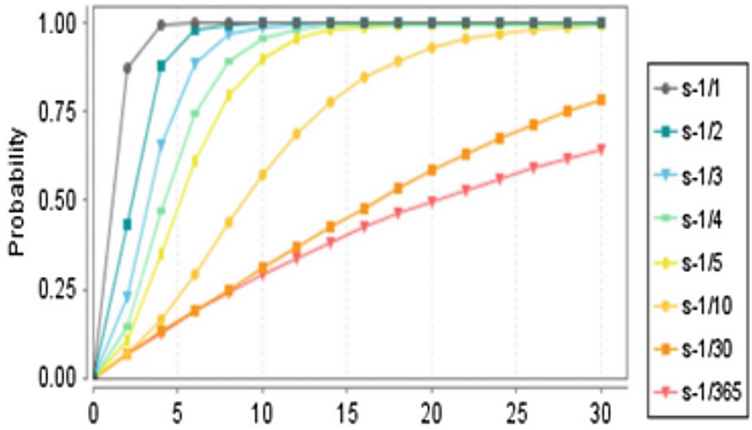

Fig. 11. The performance of the whole system (SDWSN) in 30 days with different failure rate of individual sensor nodes. The y-axis is the failure probability of the system; and the x-axis is the duration (30 days). The failure rate of sensor nodes is represented in the right; e.g., "s-1/2" means that the failure rate is once per 2 days. The number of sensor nodes is 5. The failure rate of the controller is once per 30 days

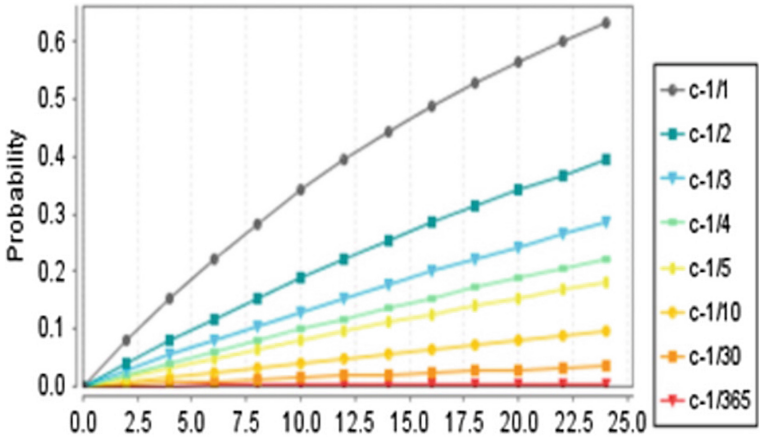

Fig. 12. The performance of the whole system (SDWSN) in 24 h with different failure rate of one controller. The y-axis is the failure probability of the system; and the x-axis is the duration (24 h). The failure rate of the single controller is represented in the right; e.g., "c-1/2" means that the failure rate is once per 2 days. The number of sensor nodes is 5. The failure rate of the sensor node is once per 30 days.

- From Figs. 12 and 13, we can see that reducing the failure rate of each controller can enhance the system reliability.
- From Figs. 12 and 14, we can see that increasing the number of controllers can also enhance the system reliability.

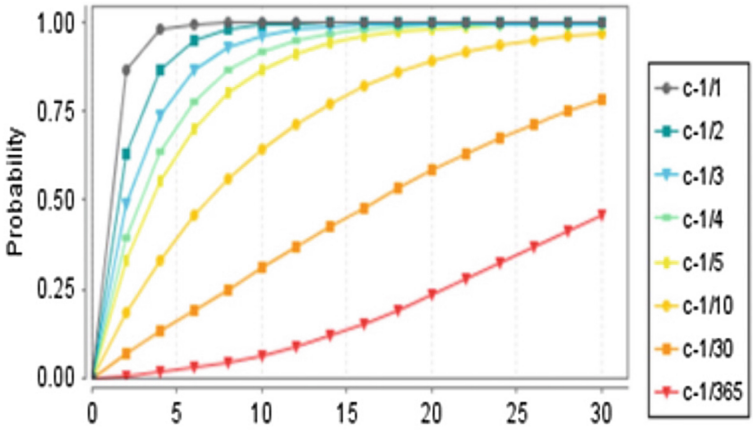

Fig. 13. The performance of the whole system (SDWSN) in 30 days with different failure rate of one controller. The y-axis is the failure probability of the system; and the x-axis is the duration (30 days). The failure rate of the single controller is represented in the right; e.g., "c-1/2" means that the failure rate is once per 2 days. The number of sensor nodes is 5. The failure rate of the sensor node is once per 30 days.

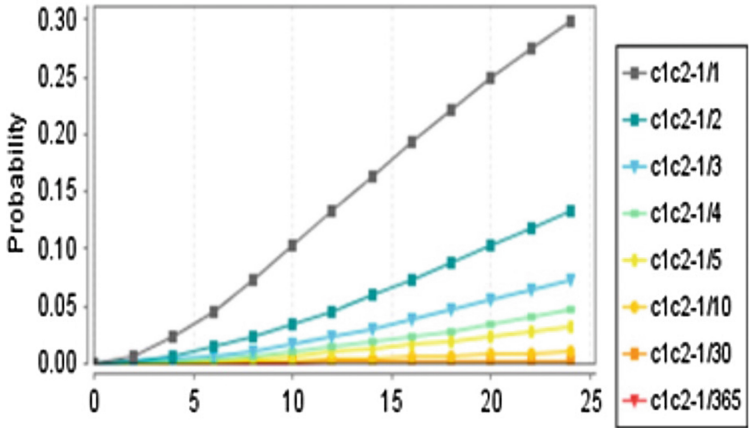

Fig. 14. The performance of the whole system (SDWSN) in 24 h with different failure rate of the two controllers. The y-axis is the failure probability of the system; and the x-axis is the duration (24 h). The failure rate of the two controllers is represented in the right; e.g., "c1c2-1/2" means that the failure rate is once per 2 days. The number of sensor nodes is 5. The failure rate of the sensor node is once per 30 days.

6 Conclusion

In this paper, to verify the reliability of SDWSN, a probabilistic model has been realized based on CTMC using PRISM. After experiments, the results illustrate the impacts of controllers and sensors on the network performance.

Firstly, increasing the number of controllers and sensors can enhance the system reliability. Secondly, the lower failure rate of controllers and sensors makes the network more dependable. Moreover, these two factors are more cognizable in long term.

Nevertheless, the limitation of the assumptions inevitably results in obstacles to find more potential correlation among components in SDWSN. For example, the relationship between the controller and sensor is omitted and this will be considered in the further research.

Acknowledgment. This work has been supported in part by the XJTLU RDF140243, by the Natural Science Foundation of Jiangsu Province under Grant BK20150376, and by the Suzhou Science and Technology Development Plan under Grant SYG201516.

References

1. Ali, S.T., Sivaraman, V., Radford, A., Jha, S.: A survey of securing networks using software defined networking. IEEE Trans. Reliab. **64**(3), 1086–1097 (2015)
2. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Model-checking continuous-time Markov chains. ACM Trans. Comput. Logic (TOCL) **1**(1), 162–170 (2000)
3. De Mil, P., Jooris, B., Tytgat, L., Hoebeke, J., Moerman, I., Demeester, P.: snapMac: a generic MAC/PHY architecture enabling flexible MAC design. Ad Hoc Netw. **17**, 37–59 (2014)
4. Ding, A.Y., Crowcroft, J., Tarkoma, S., Flinck, H.: Software defined networking for security enhancement in wireless mobile networks. Comput. Netw. **66**, 94–101 (2014)
5. Granelli, F., Gebremariam, A.A., Usman, M., Cugini, F., Stamati, V., Alitska, M., Chatzimisios, P.: Software defined, virtualized wireless access in future wireless networks: scenarios and standards. IEEE Commun. Mag. **53**(6), 26–34 (2015)
6. Han, Z.J., Ren, W.: A novel wireless sensor networks structure based on the SDN. Int. J. Distrib. Sens. Netw. **2014** (2014)
7. Harrop, P., Das, R.: Wireless sensor networks 2010-2020. Networks **2010**, 2020 (2010)
8. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: a tool for automatic verification of probabilistic systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 441–444. Springer, Heidelberg (2006)
9. Hunkeler, U., Lombriser, C., Truong, H.L., Weiss, B.: A case for centrally controlled wireless sensor networks. Comput. Netw. **57**(6), 1425–1442 (2013)
10. Jacobsson, M., Orfanidis, C.: Using software-defined networking principles for wireless sensor networks. In: 11th Swedish National Computer Networking Workshop (SNCNW), 28–29 May 2015, Karlstad, Sweden (2015)
11. Jagadeesan, N.A., Krishnamachari, B.: Software-defined networking paradigms in wireless networks: a survey. ACM Comput. Surv. (CSUR) **47**(2), 27 (2014)
12. Kirkpatrick, K.: Software-defined networking. Commun. ACM **56**(9), 16–19 (2013)
13. Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic model checking in practice: case studies with PRISM. ACM SIGMETRICS Perform. Eval. Rev. **32**(4), 16–21 (2005)
14. Lin, Y.-D., Lin, P.-C., Yeh, C.-H., Wang, Y.-C., Lai, Y.-C.: An extended SDN architecture for network function virtualization with a case study on intrusion prevention. IEEE Netw. **29**(3), 48–53 (2015)

15. Luo, T., Tan, H.-P., Quek, T.Q.S.: Sensor openflow: enabling software-defined wireless sensor networks. IEEE Commun. Lett. **16**(11), 1896–1899 (2012)
16. Menshikov, M., Petritis, D.: Explosion, implosion, and moments of passage times for continuous-time Markov chains: a semimartingale approach. Stoch. Process. Appl. **124**(7), 2388–2414 (2014)
17. Oualhaj, O.A., Kobbane, A., Sabir, E., Ben-othman, J., Erradi, M.: A ferry-assisted solution for forwarding function in wireless sensor networks. Pervasive Mob. Comput. (2015)
18. Xu, R., Huang, X., Zhang, J., Lu, Y., Wu, G., Yan, Z.: Software defined intelligent building. Int. J. Inf. Secur. Priv. (2016)
19. Steiner, R.V., Mück, T.R., Fröhlich, A.A.: C-MAC: a configurable medium access control protocol for sensor networks. In: 2010 IEEE Sensors, pp. 845–848. IEEE (2010)
20. Van Dam, T., Langendoen, K.: An adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, pp. 171–180. ACM (2003)
21. Wood, T., Ramakrishnan, K.K., Hwang, J., Liu, G., Zhang, W.: Toward a software-based network: integrating software defined networking and network function virtualization. IEEE Netw. **29**(3), 36–41 (2015)
22. Xie, J., Guo, D., Zhiyao, H., Ting, Q., Lv, P.: Control plane of software defined networks: a survey. Comput. Commun. **67**, 1–10 (2015)
23. Lu, Y., Huang, X., Huang, B., Xu, W., Zhang, Q., Xu, R., Liu, D.: A study on the reliability of software defined wireless sensor network. In: 2015 IEEE International Conference on Smart City (IEEE Smart City 2015), Chengdu, China, 19–21 December 2015