Regular Article

State Space Reduction on Wireless Sensor Network Verification Using Component-Based Petri Net Approach

Khanh Le, Thang Bui, Tho Quan

Ho Chi Minh City University of Technology, Vietnam National University, Ho Chi Minh City, Vietnam

Correspondence: Thang Bui, thang@cse.hcmut.edu.vn

Communication: received 10 December 2015, revised 18 March 2016, accepted 4 April 2016
Online publication: 10 October 2016, Digital Object Identifier: 10.21553/rev-jec.138
The guest editor coordinating the review of this article and recommending it for publication was Dr. Tran Manh Ha.

***Abstract*– With the recent advancement of Internet of Things, the applications of Wireless Sensor Networks (WSNs) are increasingly attracting attention from of both industry and research communities. However, since the deployment cost of a WSN is relatively large, one would want to make a logic model of a WSN and have the model verified beforehand to ensure that the WSN would work correctly and effectively once practically employed. Petri Net (PN) is very suitable to model a WSN, since PN strongly supports modeling concurrent and ad-hoc systems. However, verification of a PN-modeled system suffers from having to explore the huge state space of the system. In order to overcome it, in this paper we suggest a novel component-based approach to model and verify a PN-modeled WSN system. First of all, the original WSN system is divided into components, which can be further abstracted to reduce the model size. Moreover, when verifying the corresponding PN model produced from the abstracted WSN, we introduce a strategy of component-based firing, which can reduce the state space significantly. Compared to typical approach of PN-based verification, our method enjoys an impressive improvement of performance and resource consuming, as depicted in our experimental results.**

***Keywords*– Wireless Sensor Networks, Congestion Detection, Formal Verification, Component-based Petri Net.**

1 Introduction

Petri net (PN) is particularly well-suited for modeling and analyzing network systems and protocols [1]. Basically, a Petri net is a directed bipartite graph, featuring *transitions* and *places*. Each place contains a number of *tokens*. When all source places of a transition have enough requested tokens, this transition is *firable*. Whenever the transition is *fired*, the requested tokens are removed from the source places and new tokens are created in destination places.

Research on modeling network using PN have been carried out decades ago. In [2] a model is proposed to implement ATM (Asynchronous Transfer Mode) networks and the applications running over it. Transferring all multimedia data over the switching network of an ATM network constitutes a big challenge. Since ATM is a connection-oriented protocol, the switching network must establish a virtual connection from one of its input ports to an output port before forwarding incoming ATM cells (packets) along that virtual connection. Because of restricted bandwidth, multimedia data must be split to fix-length units before sending without losing behavior. Most approaches use queue-based or stochastic-based models, but however ignore synchronization. Dividing the ATM network into some synchronous models and forcing them to cooperate increases the transmission rate. Thus, using Petri nets for modeling has proved promising. [3] also uses Petri nets to model a LAN switched network architecture. Components of this model include switches, servers, clients and interactions between them. The purpose of this model is to verify the influence of the switch buffer size and the rate of packet loss on the quality of the transmission.

Recently, PN has been used to model wireless sensor networks (WSNs) [4]. WSN has been pursued by great attentions from research community due to its applications in reality including environmental monitoring, military monitoring and healthcare, etc. In [4], a model of WSN is proposed a set of sub-models including sensor models and channel models. Sensors are divided into three kinds of models, including *source*, *sink* and *intermediate*, while channels are divided correspondingly to *unicast*, *multicast* and *broadcast* models.

The next process after modeling is property verification. The verification of a property is done typically by using *Model Checking* (MC) [5]. MC allows analyzing complex concurrent systems in order to investigate their behaviors and verify whether a specified property holds during the system execution. This task is achieved by exploring the state space, i.e. a directed graph whose nodes represent reachable states of the system and the arcs represent the transitions between states. This graph is called *reachability graph*. For instance, [6, 7] use MC to verify the confidence of security protocols *i.e.* to check that a protocol has the right properties as dictated by the requirements of the corresponding system.

Besides the well-known advantages, MC suffers from the infamous disadvantage of *state space explosion*. This drawback becames serious when this technique is applied for industrial systems verification. To cope with

it, there are two main approaches proposed, including *reducing the size of model* (*i.e.* the lesser number of places and transitions, the lesser explored states of reachability graphs); or *reducing directly the states of reachability graph* during the verification process.

Our paper is the combination of two above approaches. In one hand, we divide the original model into components, each of which can be abstracted to reduce the model size. In the other hand, we suggest an approach known as *Component-Based Firing* to directly reduce the state space of rechability graph once verified. WSN model is made from two *components* being *sensor component* and *channel component*. Every component is a PN model *i.e.* it contains the set of places and transitions. This work has been adopted from [4]. However, the most interesting thing of this paper is that state space of reachability graph is smaller than the traditional one by applying our proposed method of *Component-Based Firing* that enforces the firing of the whole component at once instead of the multiple firing of each single transition. The state space is hence reduced significantly since we can skip the intermediate places and transitions on each component. As compared to [4], we enjoy much performance improvement in our experiments.

The rest of the paper is organized as follows. Section 2 recalls some background of Petri Net. Section 3 illustrates how to model and verify WSN by PN. Section 4 explains in details how the Component-Based Firing works. More extensive experiments are reported in Section 5. Subsequently, Section 6 discusses related works. Finally, Section 7 draws conclusions of our work.

2 Preliminaries

Petri net (see *e.g.* [8–10]) is a graphical formal modeling language widely used for modeling network systems and protocols [1]. There are several kinds of Petri net. In this paper, we adopt the most basic form, known as Place/Transition Net, whose formal definition is given as follows.

Definition 1 (Petri net).
*A **Petri net** is a tuple $\mathcal{N} = \langle P, T, F, W, M_0 \rangle$ where:*

- *P is a finite set of **places**;*
- *T is a finite set of **transitions**;*
- *the sets P and T are disjoint, i.e. $P \cap T = \emptyset$;*
- *$F \subseteq (P \times T) \cup (T \times P)$ is the **flow** relation;*
- *$W : F \to \mathbb{N}^+$ is the **arc weight** mapping; and*
- *$M_0 : P \to \mathbb{N}$ is the **initial marking**, representing the initial distribution of tokens.*

For example, Figure 2(a) depicts a simple PN, consisting of three places of *input*, *int* and *output*. This PN has two transitions including *generate packet* and *send packet*. All arcs of this PN are implicitly weighted by 1. This PN models a *sensor*. As its initial marking, the PN has one token in place *input*.

Definition 2 (Firing).
Let $\mathcal{N} = \{P, T, F, W, M_0\}$ be a Petri net and $M : P \to \mathbb{N}$ be a marking of $\mathcal{N}$.

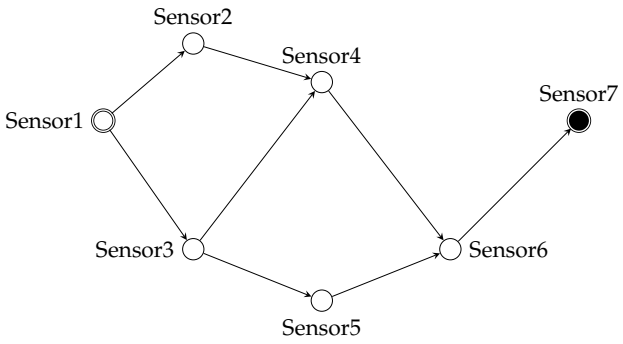


Figure 1: WSN example.

*A transition $t \in T$ is **enabled** in marking M, written $M \xrightarrow{t}$ iff $\forall p \in P : M(p) \geqslant W(p,t)$.*
*An enabled transition can **fire**, producing the successor marking M', written $M \xrightarrow{t} M'$, where: $\forall p \in P : M'(p) = M(p) - W(p,t) + W(t,p)$.*

For example, in the initial marking of the PN given in Figure 2(a), transition *generate packet* is enabled, whereas *send packet* is not. It implies that in this current status, the sensor can generate a packet but cannot send one (since there currently is no generated packet).

When *generate packet* fires, the number of tokens in *input* decreases by 1 (*e.g.* 0), whereas that of the tokens in *int* is increased by 1 (*e.g.* 1), introducing a new marking. In this new marking, *send packet* can fire, indicating that now the sensor can send a packet, which was previously generated.

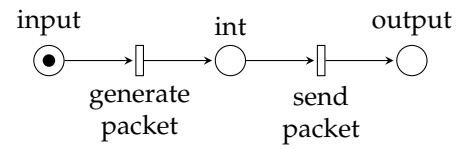
Thus, PNs are very useful and suitable to model operations of WSN, as further discussed in the following sections.

3 Model and Verification of Wireless Sensor Networks Using Petri Net

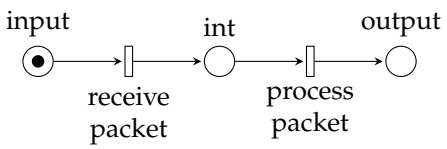
In this section, we review the idea of [4], which includes the process of modeling WSN using PN, the abstraction technique applies to PN model and PN traditional verification process.

3.1 Component-Based PN Model

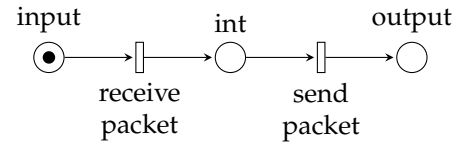
A WSN W can be defined as the set of sensors and channels, *i.e.* $W = \{S, C\}$ where S is the set of sensors and C the set channels. Depending on the function, sensors are divided into three types of *source*, *sink* and *intermediate*. A source undertakes the role of generating new packets and sending them out. Packets are received by intermediate sensors and continue transmitting via network until reaching sink. A sink receives packets and process the information on them. These sensors can be connected in *unicast*, *multicast* or *broadcast* mode, each of which specifies whether certain pairs of sensors can exchange information or not. If two sensors can communicate, we say that there is a *channel* established for these sensors. Information on sensors and channels forms the topology of a WSN. Figure 1



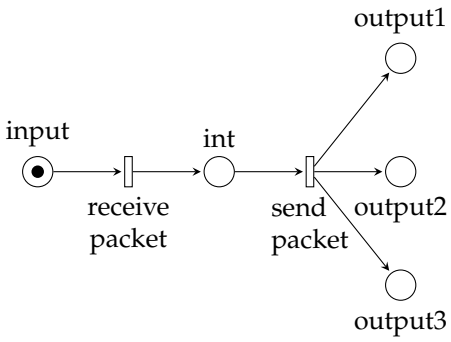
(a) Source node



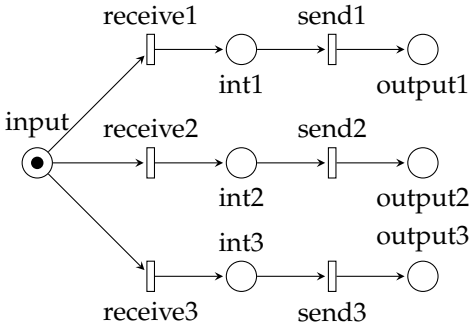
(b) Sink node



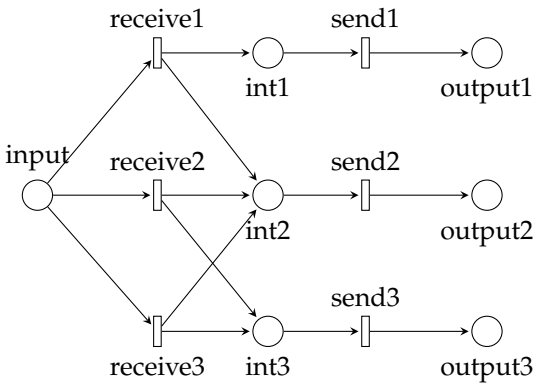
(c) Intermediate node



(d) Broadcast channel



(e) Unicast channel



(f) Multicast channel

Figure 2: Corresponding component Petri net models of sensors and channels.

gives an example of simple WSN in which Sensor 1 and Sensor 7 play the roles of *source* and *sink*, respectively.

Definition 3 (Component).
*A **component** Com is a Petri net where the set of places is the union of three disjoint sets:* $P_{Com} = P_{in} \uplus P_{out} \uplus P_{inter}$ *defined as follows:*

- P_{in} *is a non-empty set of **input places**, i.e.* $(T \times P_{in}) \cap F = \emptyset$.
- P_{out} *is a non-empty set of **output places**, i.e.* $(P_{out} \times T) \cap F = \emptyset$.
- P_{inter} *is the set of all other (intermediate) places in Com.*

For instance, a source sensor in a WSN can be modelled as a component as shown in Figure 2(a) where *Input* is the input place, *Output* the output place and *int* an intermediate place. Transition *Generate Packets* allows the *Sensor* to generate new packets and then send them with transition *Send Packets*. Similarly, a sink and intermediate sensors are modeled in Figures 2(b,c). All *Channels* have two main functions: receiving packets (transition *Receive Packets*) and sending packets (transition *Send Packets*) which are modeled as presented in Figures 2(d,e,f), corresponding to three cases of unicast, multicast and broadcast modes, respectively.

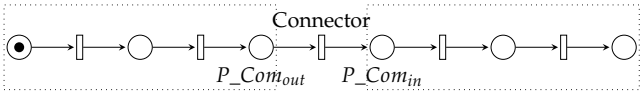


Figure 3: Connector.

Definition 4 (Component-based Petri net).
*A **component-based Petri net** is a Petri net made from a disjoint set of components* $S_{Com} = \{Com_1, \ldots, Com_n\}$ *and a set of **connectors*** $S_C = \{C_1, \ldots, C_k\}$ *where a connector* C_i *is a transition from the output places of a component to the input places of another component, as depicted in Figure 3.*

Figure 4 is an example of a Component-based Petri net which includes three components *Source* (S_1), *Channel* (T_1) and *Sink* (S_2) and two connectors T1_con and T2_con. We hereafter refer to this Petri net as *WSNCom*.

Figure 5 is a conversion from Figure 1 into PN model.

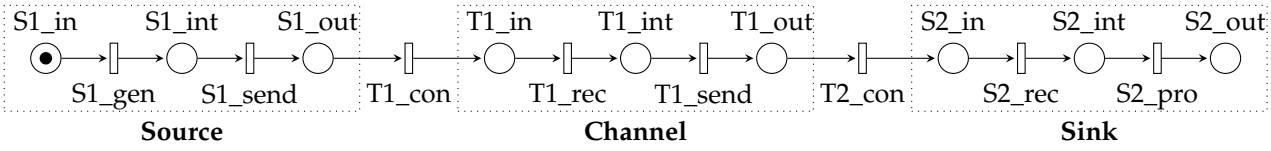


Figure 4: Component-based PN of wireless sensor network.

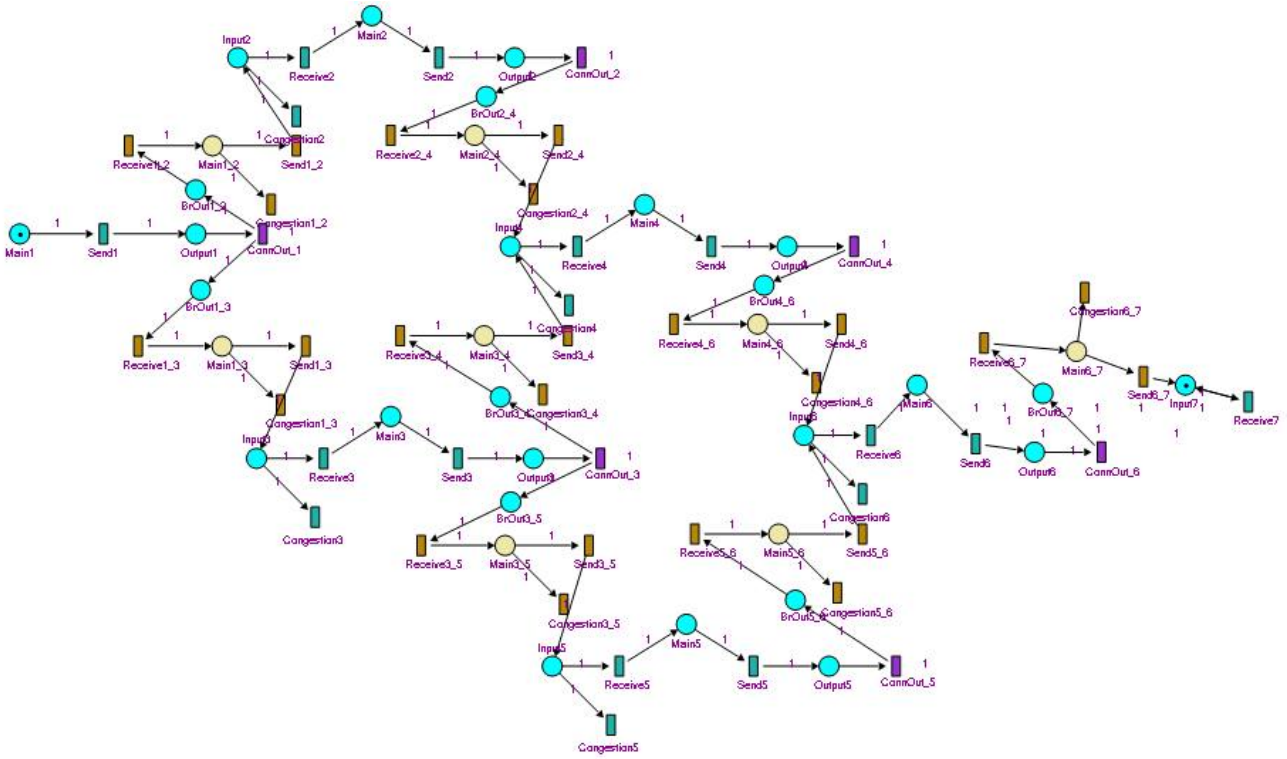
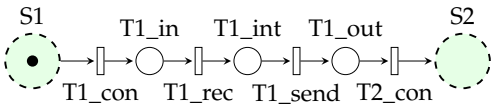


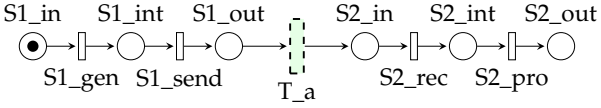
Figure 5: PN model of Figure 1.

Definition 5 (Component Abstraction). *A component Com in a Component-based PN can be reduced to an* ***abstracted place*** *(****place-based abstraction****). Alternatively, Com and all of its attached connectors can be reduced to an* ***abstracted transition*** *(****transition-based abstraction****).*

Figures 6(a,b) are examples of *Sensor Abstraction* and *Channel Abstraction*. Figure 7 is an example of PN model of Figure 1 when we abstract both sensors and channels. By abstracting a component-based Petri net model, we can obtain a new model of far smaller size, which can subsequently be used for further verification in the next step.



(a) Sensor Abstraction.



(b) Channel Abstraction.

Figure 6: Sensor and channel abstraction models.

3.2 Traditional Petri Net Verification

Once modeled properly by component-based PN, a WSN can be verified for various properties. In this paper, we target one of most important properties which is always required to be checked on a WSN: network congestion [11, 12]. We apply MC technique that is widely used for verification with PN models. This technique verifies whether a property holds on a model by exploring all of possible state space, i.e. a directed graph whose nodes represent reachable states

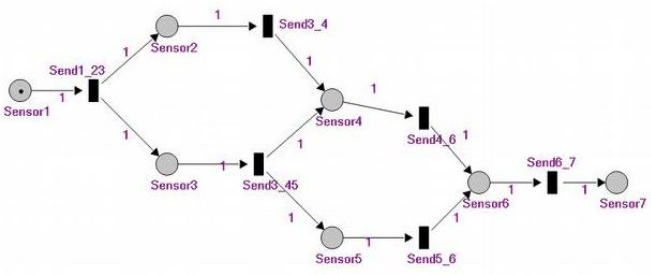


Figure 7: PN abstraction model of Figure 1.

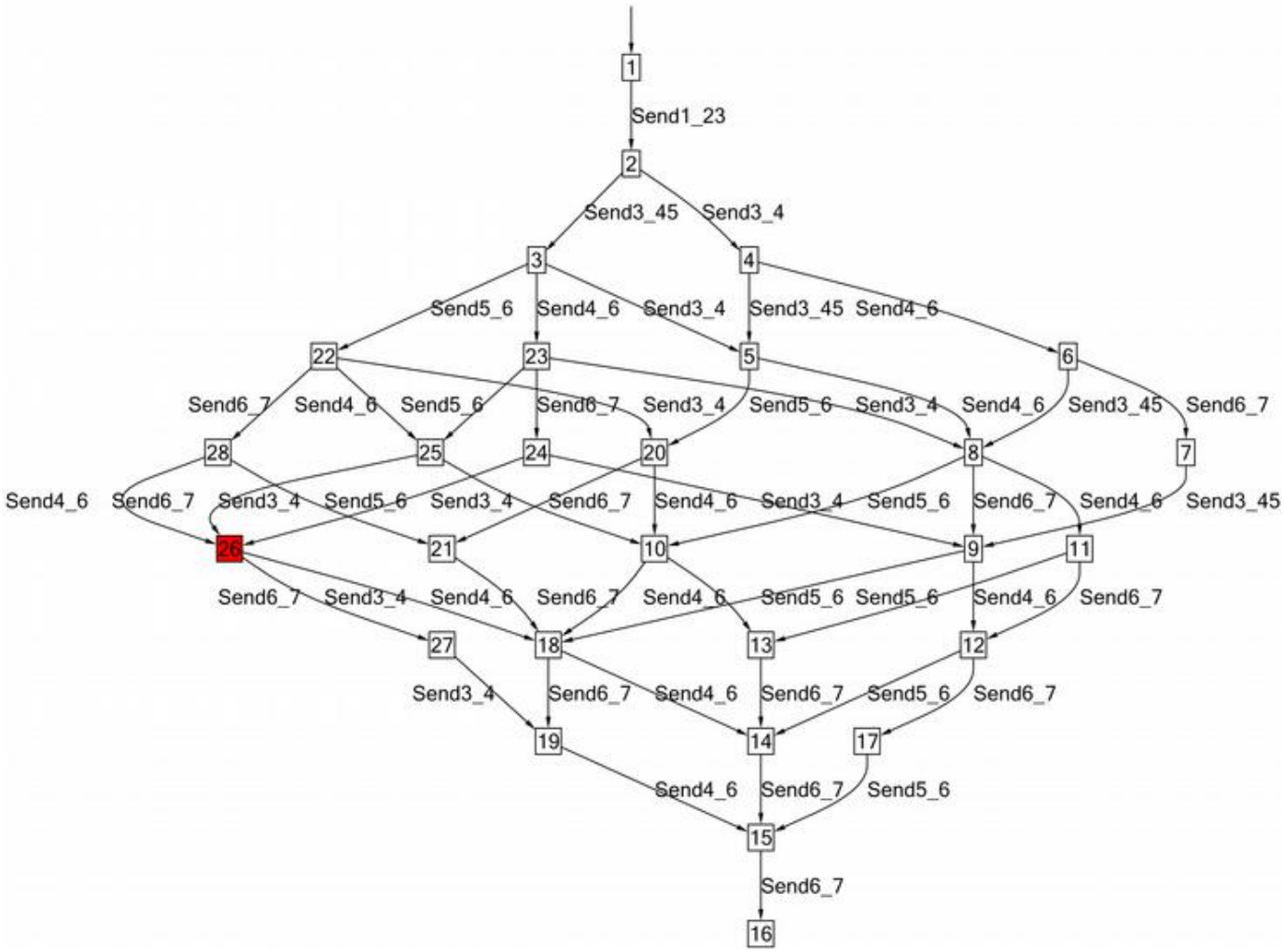


Figure 8: Generated reachability graph when using the traditional firing rule.

of the system and the arcs represent the transitions between states. This graph is called *reachability graph*.

Whenever one transition on PN is fired, a new marking is created. The set of marking forms a reachability graph. In order to check whether properties are satisfied, MC uses a simple (Depth-First or Breadth-First) Search to explore this graph from the initial state and try to reach some targets. For instance, Figure 8 is the reachability graph of traditional firing which is generated from Figure 7.

In this graph, the start state is 1, the ended state is 16 and congested state (target) is 26 (red state). Normally, the search algorithm starts from the *state* 1 and tries to reach the target state, *i.e. state* 26. A path is randomly chosen. If even though visiting ended state, the target cannot be reached on such path, the algorithm will return and choose another one. A path $1 \rightarrow 2 \rightarrow 4 \rightarrow \cdots \rightarrow 16$ is an example of unreachable target paths. It is easy to recognize that there are a huge number of paths which do not lead to target in reachability graph, thus leading to the state space exploration situation.

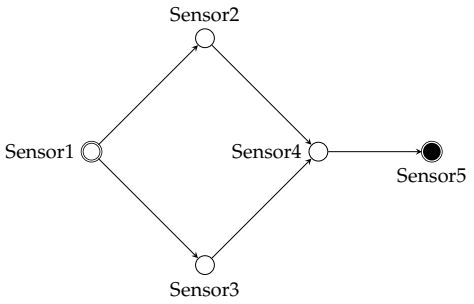


Figure 9: Another WSN example.

4 Component-Based Verification of PN-modeled Systems

In this section, we introduce *Component-Based Firing* rule which is effectively used for verification on Component-Based PN Model. In order to easily follow the idea of *Component-Based Firing*, let us start with a simple WSN topology in Figure 9 which is a WSN whose corresponding sensor-abstracted PN model is given in Figure 10.

Typically, the traditional way to model-check a PN model is to explore all markings of the model, each of

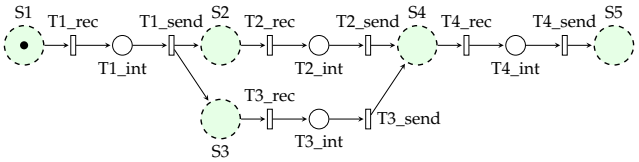


Figure 10: PN generation in broadcast mode (sensor-abstracted) of Figure 9.

which is treated as a state. However, in the case of WSN verification, we need to ensure that the WSN model works properly in terms of *timing*. The marking given in Figure 11(a) presents a situation when Sensor 1 sends packets to Sensor 2 and Sensor 3 in broadcast mode. From here, several possible markings can be generated. In Figure 11(b) is the marking presenting the situation that Sensor sends packets to Sensor 4, and then Sensor 4 further forwards the packets to Sensor 5 as illustrated in Figure 11(c).

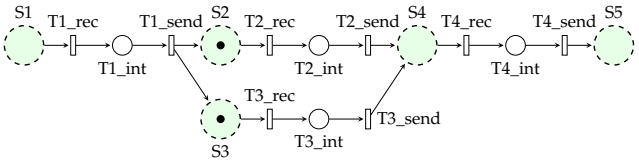
However, in the real situation, as Sensor 2 and Sensor 3 received packets from Sensor 1 and then continue sending those packets to Sensor 4 at almost the same time, Sensor 4 should only send packets to Sensor 5 after receiving packets from both Sensor 2 and Sensor 3. In other words, the marking introduced in Figure 11c is not feasible and should not be verified.

As a PN model is composed of several components, we deal with this situation by enforcing the concurrent mechanism as follows. At a certain state corresponding to a marking, a new state is introduced by *firing all of currently-enabled transitions in all components*. This simulates the real operational mechanism that all components are working concurrently in the real situation. Thus, at the marking presented in Figure 11(a), as there are two enabled transitions in two channels (note that channels are the only remained components on the model after the sensors are abstracted) connecting Sensor 2 and Sensor 3 to Sensor 4, both transitions are then needed to be fired to introduce a new state corresponding the marking illustrated in Figure 11(e). The order for firing these transitions thus does not matter. After that, Sensor 4 continues sending packets to Sensor 5, introducing a new state as depicted in Figure 11(f).

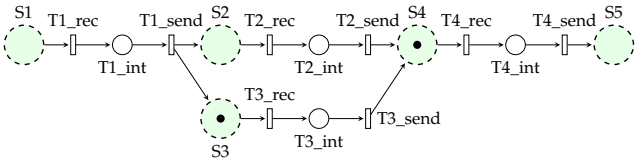
Then, we can formalize our proposed Component-based Firing as follows.

Definition 6 (Component-Based Firing).
Let $WSNCom = \{Com_1, Com_2, .., Com_n\}$ *be a Component-based Petri Net. Then, WSNCom is* Component-based Fired *at a certain marking if all enabled transactions in all components* Com_i *are fired simultaneously, immediately introducing a new marking.*

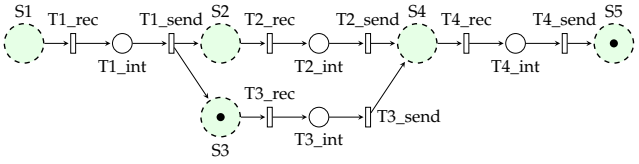
Figure 12 is the corresponding reachability graph of Figure 7 which is generated by *Component-Based Firing*. Comparing to Figure 8, we can easily observe that the approach of Component-based Firing significantly helps to reduce almost *infeasible paths* on reachability graph, which are *paths that never occur in real situation*.



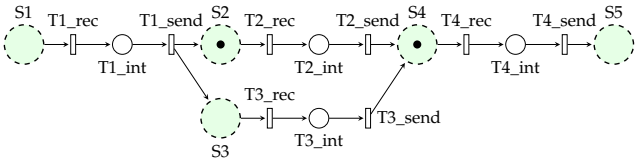
(a) Marking when Sensor 1 sends packets to Sensor 2 and Sensor 3 simultaneously (feasible marking)



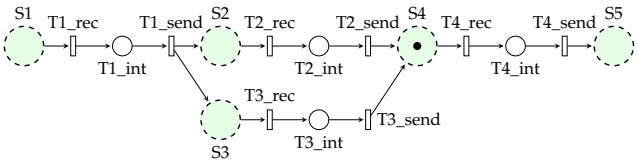
(b) Marking when Sensor 2 sends packets to Sensor 4 (feasible marking but not introducing new state)



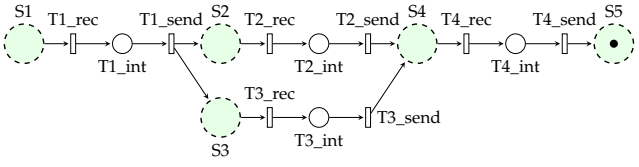
(c) Marking when Sensor 4 sends packets to Sensor 5 before receiving packets from Sensor 3 (infeasible marking since such a situation should not occur in a real WSN)



(d) Marking when Sensor 3 sends packets to Sensor 4 (feasible marking but not introducing new state)



(e) Marking when Sensor 4 received packets from both Sensor 2 and Sensor 3 (feasible marking introducing new state since all of enabled transitions in each component have been fired)



(f) Marking when Sensor 4 sends packets to Sensor 5 after receiving packets from both Sensor 2 and Sensor 3 (feasible marking introducing new state)

Figure 11: Markings of Figure 9 in broadcast mode (sensor abstraction).

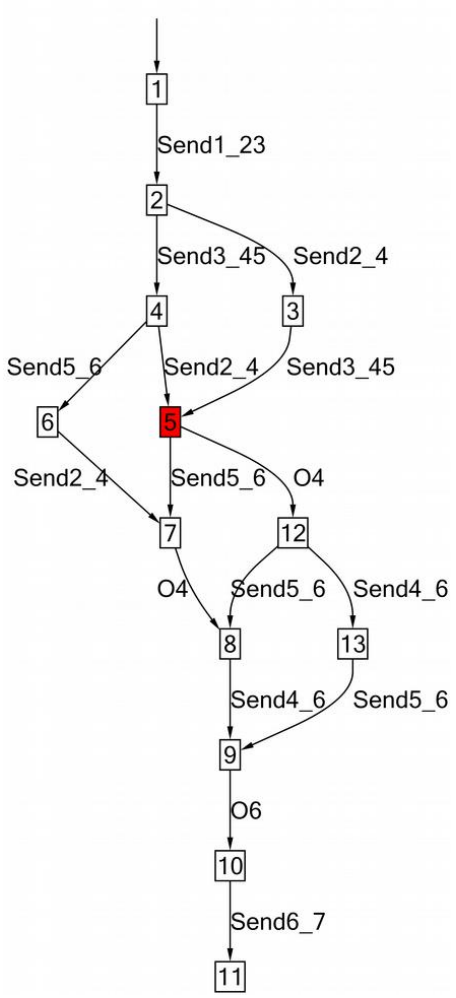


Figure 12: Generated reachability graph when using the component-based firing rule.

5 Experiments

We conducted experiments to demonstrate the efficiency of our *Component-Based Firing* in the context of congestion verification on a PN-modeled WSN. Our Component-based approach can significantly reduce the state space of congestion verification. In our experiments, the numbers of sensors range from 5 to 20. The parameters of these sensors are set to enforce the congestion.

We also verify another property *deadlock-free* of the modeled WSN. For congestion checking, we separately verify the properties of *chk-sensor-congestion* and *chk-channel-congestion* (checking whether the congestion occurs in Sensors or Channel or not).

Experimental results of *Component-Based Firing* are shown in Table I. In this experiment, we compare our firing approach to *traditional firing* on the same models. The result of *traditional firing* is extracted from the previous study [4]. We set *Timeout* as 5 hours if the system cannot terminate properly after this duration.

The results show that in most cases, the *Component-Based Firing* approach requires less visited states, as compared to the *Traditional Firing* one. It reduces the visited states from 24,476,230 to 1,223,020 (20 times) in the best case, and from 398 to 270 (1.5 times) in the worst case. Moreover, our approach can still manage very large models that cannot be handled by Traditional Firing approach (indicated by Timeout in Table I).

6 Related Works

When a system is modeled by a Petri net, its properties can be verified using model checking techniques. However, the biggest disadvantage of this approach is the infamous *state space explosion* problem. As discussed, the two major approaches of *reducing the size of model* and *reducing directly the states of reachability graph* are proposed in literature to handle this problem.

6.1 Reduction of the model size

There are two usual ways to reduce the size of model including *decomposition* and PN abstraction, which is also considered as *compositional verification*.

In the first approach, the architecture of the original system can be divided into sub-systems. The strategy of verification for this system is thus based on sub-system verification [13]. The general properties of the system can be proved based on the properties of its sub-systems. The verification process includes the following steps: verification of sub-systems, analysis on the relationship of sub-systems, combining the local properties of all sub-systems to global properties for the whole system.

Modular verification (*e.g.* [14]) is an efficient method of *decomposition* based on the design of the system. Firstly, the state space is decomposed into two parts: sub-systems and a graph to synchronize these sub-systems. A decomposition approach [15] divides the system into concurrent sub-systems. Each sub-system has its own state space. The full state space of the system is then established by the state spaces of the sub-systems or the sub-systems' actions only. This proposal is suitable for synchronous systems.

Furthermore, *abstraction techniques* or *PN abstraction* have recently been proposed. Basically, abstraction reduces a PN model into a new one which has a smaller size in terms of number of places and transitions. As a trade-off, the abstracted model cannot preserve all properties from the original one, but only some needed properties [16–18]. Thus, we cannot ensure the soundness nor completeness when verifying the abstracted model against certain properties. For instance, in the work of [19], if the abstracted model satisfies a property *e.g. deadlock*, the original model satisfies it too, but the converse is not true. Other proposals [20, 21] use counter-examples as a key for compositional verification of a system combined from multiple modules. If a counter-example is found in an abstracted module, this counter-example will be verified again on the corresponding concrete module. Otherwise, *i.e.* no counter-example found on any abstracted module for a given property, the real system does indeed satisfy the property.

Table I: Experimental results

Number of Sensors	Number of Packets	Buffer	Model	Property	Traditional Firing		Component-Based Firing	
					Visited State	Result	Visited State	Result
5	60	70	No Abstraction	*deadlockfree*	663,836	Valid	120,326	Valid
				chk-channel-congestion	2,546,821	Not valid	86,321	Not valid
				chk-sensor-congestion	2,546,779	Not valid	87,360	Not valid
			Channels Abstraction	*deadlockfree*	18,304	Valid	7,413	Valid
				chk-sensor-congestion	48,131	Not valid	21,031	Not valid
			Sensors Abstraction	*deadlockfree*	21,504	Valid	12,365	Valid
				chk-channel-congestion	81,856	Not valid	65,230	Not valid
10	60	70	No Abstraction	*deadlockfree*	1,247,938	Valid	93,120	Valid
				chk-channel-congestion	394,019	Valid	289,631	Valid
				chk-sensor-congestion	21,203	Valid	10,136	Valid
			Channels Abstraction	*deadlockfree*	158,000	Valid	8,930	Valid
				chk-sensor-congestion	150,423	Valid	95,320	Valid
			Sensors Abstraction	*deadlockfree*	14,855	Valid	9,563	Valid
				chk-channel-congestion	60,413	Valid	60,231	Valid
15	60	70	No Abstraction	*deadlockfree*	Timeout	-	Timeout	-
				chk-channel-congestion	Timeout	-	856,232	Not valid
				chk-sensor-congestion	398	Valid	270	Valid
			Channels Abstraction	*deadlockfree*	Timeout	-	4,856,301	Valid
				chk-sensor-congestion	29,150	Not valid	18,652	Valid
			Sensors Abstraction	*deadlockfree*	Timeout	-	Timeout	-
				chk-channel-congestion	Timeout	-	522,014	Not valid
20	60	70	No Abstraction	*deadlockfree*	Timeout	-	30,125,693	Not valid
				chk-channel-congestion	Timeout	-	98,563	Valid
				chk-sensor-congestion	398	Valid	270	Valid
			Channels Abstraction	*deadlockfree*	74,465	Valid	65,320	Valid
				chk-sensor-congestion	24,476,230	Valid	1,223,020	Valid
			Sensors Abstraction	*deadlockfree*	501,452	Valid	403,560	Valid
				chk-channel-congestion	1,412,365	Valid	7,412,865	Valid

6.2 Direct reduction of the reachability graph states

Reducing directly the states of reachability graph were proposed to handle state space reduction such as partial order reduction (*e.g.* [22, 23]), symmetries use (*e.g.* [24, 25]) or unfolding methods (*e.g.* [26]).

Partial order reduction methods exploit this redundancy and visit only a subset of the reachable states. The main idea of this concept is replacing the full transitions system by a small subset one. Paths which have the same actions (but different searching order) of the visited others are skipped. So, the new reachability graph will be smaller than the old one.

Basically, the idea of exploiting symmetry is as follows. Given a model including *States*, *Transitions* and their *Labels*, a symmetry group G is a group which partitions the state set S into equivalence classes called *orbits*. A new quotient model MG is constructed that contains one or more representative from each orbit. The state space of the quotient model will, in general, be much smaller than the original state space S. This makes it possible to verify much larger structures.

In recent years, there has been a growing interest in use of unfolding approaches. The unfolding semantics is a branching partial order semantics which represents in a single structure for all the possible events in computations. Each branch represents a concurrent execution of the net. The unfolding provides an efficient representation of the state space of the system, not only taking advantage of a partial order representation of concurrency, but also keeping together different computations in its branching structure until a conflict is reached.

7 Conclusion

This paper presents an efficient way of reducing state space on WSN using Petri nets. Especially, we propose *Component-based Firing*, a mechanism which allows synchronously firing components on model in order to decrease the number of redundant firing transitions in each component. It thus significantly reduces the state space generated for congestion verification, as shown in our experimental results, which aimed at detection of congestion in a PN-modeled WSN.

Acknowledgment

This research is funded by Ho Chi Minh University of Technology under grant number TNCS-2015-KHMT-38.

References

[1] J. Billington, G. R. Wheeler, and M. C. Wilbur-Ham, "PROTEAN: A high-level petri net tool for the specification and verification of communication protocols," *IEEE Transactions on Software Engineering*, vol. 14, no. 3, pp. 301–316, 1988.

[2] M. Reid and W. M. Zuberek, "Timed petri net models of ATM lans," in *Application of Petri Nets to Communication Networks, Advances in Petri Nets*, 1999, pp. 150–175.

[3] D. A. Zaitsev, "Switched LAN simulation by colored petri nets," *Mathematics and Computers in Simulation*, vol. 65, no. 3, pp. 245–249, 2004.

[4] K. Le, T. Bui, T. Quan, L. Petrucci, and É. André, "Component-based abstraction of petri net models: An application for congestion verification of wireless sensor networks," in *Proceedings of the Sixth International Symposium on Information and Communication Technology, Hue City, Vietnam, Dec. 3-4*, 2015, p. 51.

[5] C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.

[6] E. M. Clarke, S. Jha, and W. R. Marrero, "Verifying security protocols with brutus," *ACM Transactions on Software Engineering and Methodology*, vol. 9, no. 4, pp. 443–487, 2000.
[7] ——, "Efficient verification of security protocols using partial-order reductions," *International Journal on Software Tools for Technology Transfer*, vol. 4, no. 2, pp. 173–188, 2003.
[8] C. A. Petri, "Kommunikation mit automaten," Ph.D. dissertation, Darmstadt University of Technology, Germany, 1962.
[9] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.
[10] K. Jensen and L. M. Kristensen, *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*. Springer, 2009.
[11] C. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: congestion detection and avoidance in sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, California, USA, Nov. 2003, pp. 266–279.
[12] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Baltimore, MD, USA, Nov. 2004, pp. 134–147.
[13] E. M. Clarke, D. E. Long, and K. L. McMillan, "Compositional model checking," in *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89)*, 1989, pp. 353–362.
[14] C. Lakos and L. Petrucci, "Modular analysis of systems composed of semiautonomous subsystems," in *Proceedings of the Fourth International Conference on Application of Concurrency to System Design (ACSD 2004)*, Hamilton, Canada, Jun. 2004, pp. 185–194.
[15] A. Valmari, "Compositionality in state space verification methods," in *Application and Theory of Petri Nets (APN 1996)*, 1996, pp. 29–56.
[16] S. Haddad, J. Ilié, and K. Klai, "Design and evaluation of a symbolic and abstraction-based model checker," in *2nd International Symposium on Automated Technology for Verification and Analysis (ATVA 2004)*, 2004, pp. 196–210.
[17] K. Klai and D. Poitrenaud, "MC-SOG: an LTL model checker based on symbolic observation graphs," in *29th International Conference on Applications and Theory of Petri Nets (PETRI NETS 2008)*, 2008, pp. 288–306.
[18] A. Duret-Lutz, K. Klai, D. Poitrenaud, and Y. Thierry-Mieg, "Self-loop aggregation product - A new hybrid approach to on-the-fly LTL model checking," in *9th International Symposium on Automated Technology for Verification and Analysis (ATVA 2011)*, 2011, pp. 336–350.
[19] K. Klai, S. Haddad, and J. Ilié, "Modular verification of petri nets properties: A structure-based approach," in *Formal Techniques for Networked and Distributed Systems (FORTE 2005)*, 2005, pp. 189–203.
[20] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement for symbolic model checking," *Journal of the ACM (JACM)*, vol. 50, no. 5, pp. 752–794, 2003.
[21] É. André, H. Ochi, K. Klai, and L. Petrucci, "A counterexample-based incremental and modular verification approach," in *17th Monterey Workshop on Development, Operation and Management of Large-Scale Complex IT Systems*, ser. Lecture Notes in Computer Science, R. Calinescu and D. Garlan, Eds., vol. 7539. Oxford, England: Springer, Aug. 2012, pp. 283–302.
[22] D. Peled, "All from one, one for all: on model checking using representatives," in *Proceedings of the 5th International Conference on Computer Aided Verification (CAV '93)*, Elounda, Greece, Jun. 28 - Jul. 1 1993, pp. 409–423.
[23] P. Godefroid, *Partial-Order Methods for the Verification of Concurrent Systems - An Approach to the State-Explosion Problem*, ser. Lecture Notes in Computer Science. Springer, 1996, vol. 1032.
[24] E. A. Emerson and A. P. Sistla, "Symmetry and model checking," *Formal Methods in System Design*, vol. 9, no. 1/2, pp. 105–131, 1996.
[25] E. M. Clarke, S. Jha, R. Enders, and T. Filkorn, "Exploiting symmetry in temporal logic model checking," *Formal Methods in System Design*, vol. 9, no. 1/2, pp. 77–104, 1996.
[26] J. Esparza, "Model checking using net unfoldings," *Science of Computer Programming*, vol. 23, no. 2-3, pp. 151–195, 1994.

Ms. Khanh Le is the Lecturer in the Faculty Information Technology of Saigon University, Vietnam. She received the Bachelor degree in Mathematics and Computing from Ho Chi Minh City University of Natural Science in 2005 and received Master degree in Computer Networking from Paris VI University in 2009. Now, she is a PhD student of Ho Chi Minh City University of Technology. Her current research in Quality of services for wireless sensor networks, System modeling and verification.

Dr. Thang Bui is currently a Lecturer in Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam, and a member of the Laboratory for Systems Analysis and VErification (SAVE), which aims at developing automated techniques for analyzing and reasoning on computer-based systems. His research focuses on Software Verification, particularly Model checking, which is to model the real world application and to check for any violation of desired properties. He holds a Bachelor degree in Computer Engineering from Ho Chi Minh City University of Technology in 1997, a Master degree in Computer Science and Engineering from the Asian Institute of Technology, Thailand, in 2001, and a PhD degree in Computer Science and Engineering from the University of New South Wales, Australia, in 2010.

Dr. Tho Quan is an Associate Professor in the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam. He received his B.Eng. degree in Information Technology from Ho Chi Minh City University of Technology in 1998 and received Ph.D degree from Nanyang Technological University, Singapore, in 2006. His current research interests include formal methods, program analysis/verification, the Semantic Web, machine learning/data mining and intelligent systems. Currently, he heads the Department of Software Engineering of the Faculty. He is also serving as the Chair of Computer Science Program (undergraduate level).