Formal Verification of CAN Bus in Cyber Physical System

Rui Wang
Beijing Key Laboratory of Light Industrial Robot and Safety Verification
Capital Normal University
Beijing, China

Yong Guan
College of Information Engineering
Capital Normal University
Beijing, China

Xiaojuan Li
College of Information Engineering
Capital Normal University
Beijing, China

Rui Zhang
Beijing Key Laboratory of Light Industrial Robot and Safety Verification
Capital Normal University
Beijing, China

***Abstract*—Cyber physical system (CPS) is a multi-dimensional complicated system integrating computing, communication and physical environment. CPS is widely used in safety-critical areas such as aerospace, intelligent transportation and medical equipment. So ensuring the security and reliability of CPS is of great significance. Formal verification is one of the useful ways. This paper builds timed automata models for the communication process of CAN bus used in CPS. Our research especially analyses the gateway in the communication process, and simulates the transmission with different rates between the external environment and internal unit. The task also takes into account the packet transmission priority. The model checking tool Uppaal is used to verify the functional and real-time properties. The verification results illustrate that the established model can meet the relevant properties, and the packet can be transmitted in an orderly and efficient manner.**

***Index Terms*—Formal verification; Timed automata; CPS; CAN bus; Communication system**

I. Introduction

CPS [1] integrating advanced sensing, computing, communication technologies manages the interactions of the human, control program and physical environments. It is a complex system requiring efficient coordination. CPS is playing an increasingly important role in our daily lives, such as medical robots, industrial automation and control Systems and self-driving cars and so on. The interaction of the CPS system with external information is executed by its communication system. So it is of importance to ensure the safety and reliability of the communication process. The Controller Area Network (CAN) bus is a serial bus and widely used CPS [2]. It contains transceivers, arbiter and gateway modules. The transceiver can ensure the smooth reception and transmission of messages. The arbiter can guarantee the correct format and priority of the message. The gateway [3] is used for message transmission in the different rates, in which the packets are repackaged and delivered. The key modules on these communication buses ensure the transmission of the data.

The traditional verification approaches of the CAN bus protocol include simulation and testing. Simulation uses the software system to simulate the CAN bus operating environment, analyse the communication process and draw conclusions. Lindsey et al. studied the IP radiation resistance and solidification of the CAN bus [4]. Kim et al. used CANoe to simulate the heavy truck communication system, which confirmed its reliability based on CAN bus communication [5]. CANoe is a simulation software with programming function, which can analyse the transmitted signal in real time. But it relies too much on hardware facilities, and the number of parameters is limited, resulting in incomplete experimental results. The testing method is generally carried out in the final stage of the design, which not only consumes a lot of time, but also has low reusability, bringing about a great waste of cost. Mansor et al. used the FMEA method to analyze the threats and vulnerabilities of the CAN bus network, realized the experimental setup of the communication network and derived the security requirements of the CAN bus [6]. Using unsupervised learning techniques, Umberto proposes a new method for analyzing and classifying driver behavior, using selected subsets of CAN bus signals for classifying driver behavior in an uncontrolled environment in near real-time [7]. Jin proposes an improved algorithm to calculate the response time of a CAN bus consisting of nodes with hardware and software buffers, and analyze the worst response time of the CAN bus [8]. Pan designed and validated a formal model of the CAN bus protocol, focusing on the arbitration process, the transmission process and the fault limiting mechanism in the model [9].

Formal verification have played a increasing important role in many fields in recent years, such as construction field and medical field [10]–[12]. It includes model checking [13] and theorem proving. In contrast, model checking [14] can automatically prove the system and reduce manual intervention. Timed automata theory is one of the crucial foundations of

This paper is sported by National Key R&D Program of China (Project No. 2019YFB1309900), NSFC 61877040 and Academy for Multidisciplinary Studies, Capital Normal University(19530012005). Corresponding author: Yong Guan,Email: Guanyong@cnu.edu.cn

978-1-7281-8915-4/20/$31.00 ©2020 IEEE
DOI 10.1109/QRS-C51114.2020.00050

model checking technology [15], which plays a key role in the formal verification of real-time systems. The idea is to use the form of timed automata to build the target system model, then search and traverse all states of the model, finally verify whether the system can satisfy the required properties. The process of model checking can be divided into three steps: firstly the properties needs to be met in the target system will be expressed with some logical formulas, such as CTL formula and LTL formula; secondly, people will abstract the target system reasonably and build a model for it, in which timed automata, Kripe structure and transfer structure can be employed in the modeling process; finally, the model checking tool is used to verify these properties. If the verification result is valid, we will realize the established model satisfies the requirement; otherwise a counterexample will be given. Under the circumstances, it is essential to check the model by the counter-example. If there is no problem in the model establishment, it is necessary to consider whether the system does not fit the bill.

The main contribution of this paper is analysing CAN bus communication system and establishing the time automaton model for each part in the message transmission process. In particular, we model and verify the gateway to ensure the message transmitted normally at high speed and low speed. Farther more, the properties of CPS that need to be satisfied in the actual operation are abstracted and formally described by the computational tree logic(CTL) formula, and the model checking tools are used to verify these properties so that we can ensure that the correct and timely data transmission when it meet the emergency situation.

This paper is organized as follows. Section II describes the formal syntax and semantics of timed automata. Section III analyses the principle of CAN bus. In section IV, the models of each communication module are established and the process of message transmission in the whole system is explained. In section V, the key properties that must be met in the communication system are abstractly explained, and the properties are formulated by CTL formula. Finally, the research is summarized in section VI, and future work of this research are proposed.

II. Timed automata

Timed automata [16] are finite automata with extension of time variables. Let V be a finite set of variables including clock variables C and data variables X, $V = C \cup X$ and $C \cap X = \emptyset$. $\psi(V)$ expresses invariant and guard formulas. We have $\psi ::= e \mid \psi \ \wedge \ \psi$, e has the form of $c \sim n$ or $x \sim n$, $c \in C, x \in X$, $\sim \in \{\leq, \geq, =, <, > \}$ and $n \in \mathbb{N}$. $v := expression$ defines the assignment operation, where $v \in V$. Let Q denote all the assignment formulas.

Definition 1 Timed automaton is a tuple $\mathcal{A}(L, l_0, A, V, I, T)$

- L: a finite set of locations
- l_0: initial locations
- A: a finite set of actions
- V: a finite set of variables
- $I : L \to \psi(V)$ a location constraint function
- T: a finite set of transitions

$E \subseteq L \times \psi(V) \times T \times Q \times L$. Each edge has a source location l, a target location l_1. When guard $g \in \psi$ is satisfied, the transition happens and a subset of variables in V are updated by formula $q \in Q$. A transition $t\langle l, g, t, q, l' \rangle$ can be written as $l \xrightarrow{g,t,q} l'$.

At first all clocks are set to 0. The automaton starts at the initial state l_0 With time passing by, the clock variables increase at the same rate satisfying the invariant constraints $I(l_0)$. The system can remain still in this location or transit to l_1 if the variables satisfy the guard g. With the transition, action a is taken and variables are updated by formula q.

Definition 2 The semantics of a timed automaton $\mathcal{A}(L, l_0, A, V, I, T)$ is defined as a labeled transition system $\mathcal{S}(\mathcal{A}) = \langle S, s_0, \rightarrow \rangle$. $S \subseteq L \times mathbbR$ is a set of states, s_0 is the initial state, $\rightarrow \subseteq S \times (U \cup A) \times S$ is the set of relations, divided into the following two cases.

- Elapses of time transitions: for $d \in \mathbb{R}^+$, $(l, u) \xrightarrow{d} (l, u + d)$, if for $\forall d' \leq d$, u and $u + d'$ satisfy $I(l)$, and
- Location switch transitions: $(l, u) \xrightarrow{a} (l', u')$, if $\exists e \ (l, a, g, r, l') \in E$, $u' = r(u)$, u satisfies guard g, $r \in U$ and u' satisfies $I(l')$.

Usually the system is build using timed automata networks $\overline{\mathcal{A}} = \mathcal{A}_1 \parallel ... \parallel \mathcal{A}_n$.

III. CAN bus in CPS

A. CAN protocol introduction

The CAN bus [17] has strong real-time performance and high reliability, and is easy to develop and expand. Therefore, it is also widely used in the automotive field, rail transit, marine and navigation.

1) Structure of CAN bus: The CAN bus is a two-wire structure twisted avoiding external radiation and electromagnetic interference. Each control unit connected to the CAN bus can send data and be able to selectively read the data . When one of the control unit fails, it will not affect others. CAN bus enables transmission at different rates, the high speed bus is used in the crucial system, and the low speed bus is used in the auxiliary system. For different speeds, it is essential to "translate" the bus connection gateways with different rates. The gateway is usually an independent control module or borrows other control modules. The CAN bus structure is shown in Figure 1:

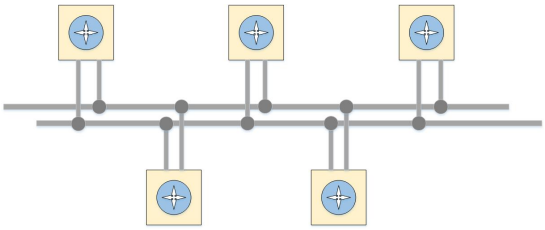

Fig. 1. Structure of CAN bus

2) Message format of CAN bus: The CAN protocol divides the message into four types: data frame, remote frame, error frame and overload frame. Each type has its fixed format. The frame format of the CAN protocol is divided into standard and extended formats. The identifier of standard format is 11 bits and the extended format is 29 bits. In this paper the model uses standard format. The data in the message is composed of the dominant bit (0) and the recessive bit (1). The dominant bit has a higher priority and the recessive bit has a lower priority.

3) Arbitration mechanism of CAN bus: The CAN bus uses non-destructive arbitration techniques. Among them, non-destructive means that the data and time are not lost, which is determined by the ID of bitwise arbitration. If the bus is idle when sending a task, any data frame can occupy the bus. After receiving the data, the unit receiving the message will send a feedback message to the sending unit, indicates that the task is completed, otherwise the message will be resent; When two or more units of different IDs transmit data at the same time, a collision occurs. At this time, the arbitrator needs to arbitrate, that is, the identifier of the packet is compared bit by bit, and the packet with the highest priority is directly sent, and the packet with the lower priority will be returned and wait to be sent again when the bus is idle. Therefore, there is no loss in the transmission time of the message with the highest priority. The priority is determined by the ID of the packet. The smaller the ID, the higher the priority.

B. CPS based on CAN bus communication

In the process of communication, the transmission process of data and commands is carried out by means of the automobile bus system. Generally, CAN bus is suitable for real-time systems, and its advantages are: few wire harnesses and light weight. A wealth of features can be implemented. At present, the most widely used bus in CPS is still the CAN bus [18]. The CPS is equipped with a wide range of sensors, such as cameras, infrared sensors and lidar sensors, which together form the CPS sensor system [19]. In the communication process, the carrier of the transmission of the message is a CAN bus system. It contains modules such as transceiver [20], arbitrator [21] and gateway. The key modules on these communication buses can ensure the correctness of the transmission process. Figure 2 shows the process of CPS communication system [22]:

IV. Formal Modelings

A. UPPAAL: a symbolic model detection tool

UPPAAL is a graphical automated model checking toolbox developed by Uppsala University and Aalborg University that uses clock variables to characterize changes of continuous time. It provides a method for deadlocks detection, uses clock variables to accurately describe the conditions of the real-time performance in the target system [23]. This study establishes a timed automata model for each key module in CAN-based communication system and imitates the entire process, which exhaustively search all possible conditions to verify the correctness of its properties.

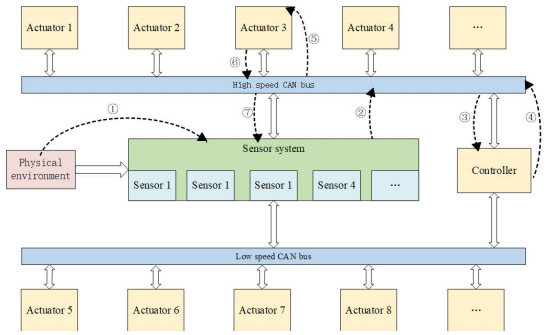

Fig. 2. Framework of communication system in CPS

TABLE I
Path Formula in Uppaal

Path formula	Expression meaning
A <> p	p will inevitable become true.
A [] p	p holds invariantly.
E <> p	it is possible to reach a state in which p is satisfied.
E [] p	p is potentially always true.
P imply q	if p satisfies, then q also satisfies.

UPPAAL consists of concurrent processes [24], and each process is established as a corresponding timed automata. For each time automaton, it contains a series of locations to represent its different states and transitions to indicate the conditions and parameters required to change state. UPPAAL can simulate a real-time system with concurrent time using a timed automata model network with bounded integer variables. Each process can be described by a parameter control structure, a clock variable, and a channel between processes. Among them, the role of the channel is to achieve synchronous communication between processes. For example, if *a* is used to represent a channel, then *a!* indicates that a synchronization signal is sent to other processes, and *a?* indicates that a synchronization signal from another process has been received. In this tool, asynchronous communication is also possible through shared variables. It uses the Backus-Naur Form grammar to describe the properties of the target system. This grammar simplifies the CTL formula, including path formulas and state formulas. The path formula is a description of the path or trajectory in the model. The state formula is in the model. The status is described. Table I shows several common path formulas:

B. Establishment of communication system model

The communication process of the CPS can be described as the following steps: (1)the sensor system transmits the collected data to the transceiver in the CAN bus, and waits for arbitration; (2)the arbiter arbitrates the data in each frame of the message bit by bit and send them to the transceiver according to the priority; (3)the gateway determines whether the message requires transmitting at a variable speed, and sends the signal to the transceiver after the speed is consistent; (4)after receiving the feedback message, the transceiver transmits

the message to the controller of the system; (5)the controller sends the command to each sub-control module via the bus system after analyzing the content of the message; (6)each actuator module performs its own operations and performs corresponding operations after receiving the message; (7)after the execution, each actuator will send feedback information to the sensor system. By this end of the communication process, a new round of transmission can be started.

In accordance with the whole procedure of CPS communication from data acquisition to processing, our research makes a reasonable abstraction of the CPS communication system as shown in Figure 3, paying attention to the communication process of the vehicle and ignoring other unrelated factors. The system is divided into sensor system, controller, actuator and CAN bus system, wherein the CAN bus system can be divided into four parts: bus, transceiver, arbiter and gateway. The formal modeling and analysis of these parts can further clarify the relationship between the key modules of the communication process, ensuring the accuracy and reliability of the model.

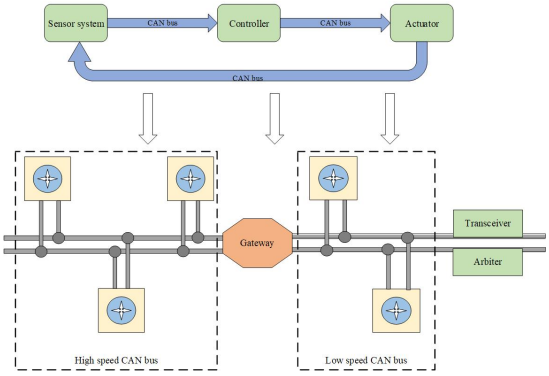

Fig. 3. Framework of abstract model

According to the definition of the CAN bus, there are multiple control units in CPS that are connected to the CAN bus, such as sensors and actuators. For the refinement of modeling, only one model is built for each sensor and actuator, and if necessary, it is instantiated into multiple models on the basis of the template in UPPAAL. Therefore, in the sensor model, the characteristics of multiple sensors are collected. And whether each sensor is collected the data is set to a Boolean variable. When the sensor sends data, the Boolean variable is updated to true, otherwise it is false; the characteristics of multiple executors are aggregated in the actuator, and each executor receives a command as a Boolean variable. When the executor is triggered, the Boolean variable is updated to true; otherwise, it is false.

1) Sensor model: The sensor system is an indispensable section of the communication process. Its function is to collect data of external things using different sensors and detect its own properties, while transmitting data to the controller of the CPS in time. The critical states in sensor system includes idle state, ready state, arbitration state, sending state, receiving state and finish state.

First sensor system is at the *initial* state, and the variable *begin* is set to 1 at the beginning of each transfer. After a communication cycle, *begin* is set to 0, and it is initialized to 1 at the head of the next cycle. When the message starts to transmit, the sensor system prepares to send the received data and a synchronization signal *start_data!* to the transceiver. If the CAN bus system is not occupied, it enters the ready state for transmission. At this time, the transceiver will receive the signal and use the function *writing_data()* to write each packet, where the packet includes the message ID, type, identifier, and transmission speed. In the case of successful arbitration, the packet will be migrated to the *sending* state. Once the sensor system captures an emergency during transmission, the corresponding variable is set to 1, and the transmission of one cycle will come to an end. In addition, after receiving the synchronization signal *interrupted?* send by the actuator when processing is complete, the feedback information is accepted, and the operation of each actuator will be executed, at the same time the corresponding Boolean variable is updated to true. It means that CPS has made corresponding response and reaction to the emergency situation. The model of the sensor is shown in Figure 4:

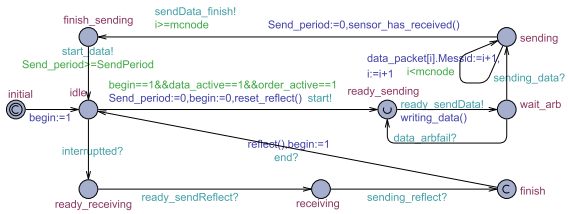

Fig. 4. Sensor model

2) Controller model: The controller undertakes the central work in the communication system. It is the intermediate module associating the sensor system and the actuator. The message data firstly is analyzed and processed, and the commands are distributed to the various actuator modules. So it is an integral part of the communication system. The critical states in controller include idle state, arbitration state, sending state, receiving state and finish state. After receiving the synchronization signal *start_data?* and the message from the sensor system, the controller starts processing and sends a command to each actuator module and writes the command packet, which will be transmitted via the transceiver after arbitration. The model of the controller is appeared in Figure 5:

3) Actuator model: Each actuator module in CPS assumes different tasks, which have been analyzed above. The critical states in controller includes idle state, receiving state, execute state, sending state and finish state. The executor starts executing the command after receiving the message. In this process, the execution time is limited in 3 time units. In the mean while, the communication system is required

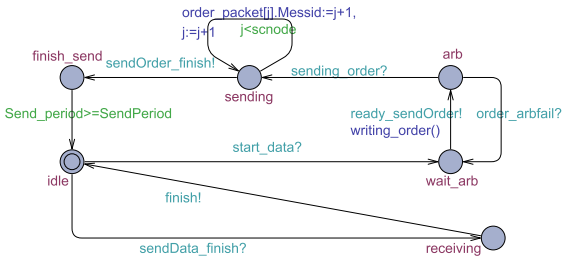

Fig. 5. Controller model

to provide sound, lighting, visual and video prompts in the event of an emergency to assist the driver in making the correct response. After the execution is completed, a feedback message is sent to the sensor system indicating that the next round of transmission can be started. And the state of the bus is changed from busy to idle. The model of the actuator is shown in Figure 6:

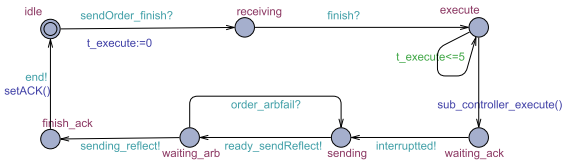

Fig. 6. Actuator model

4) Transceiver model: The transceiver first determines the state of the CAN bus. If the bus is at a busy state, that is *data_active* is 0, then the message enters a waiting state; if the bus is idle, that is *data_active* is 1, then the message enters the arbiter for priority judgment. When the message is successfully arbitrated, data transmission will begin; but if the arbitration fails, the above steps are repeated. The model of the transceiver is shown in Figure 7:

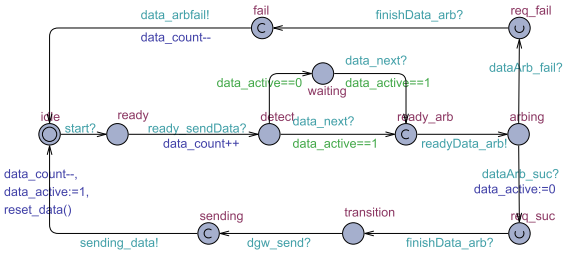

Fig. 7. Transceiver model

5) Arbiter model: After the synchronization signal *readyData_arb?* sent by the transceiver is received, the arbitration operation is started. The content of the message is calculated bit by bit, that is, whether the value of the current data bit matches the obtained bus, if not match indicates that the message transmission failed. The packet will exit from the arbitration, and the automaton migrated to the state where the request failed; if the two match, but the identifier has not been compared, then the next bit will be arbitrated; if the two match and the identifier has been completed, the message transmission is successful and the arbitration work is completed. The model of the arbiter is shown in Figure 8:

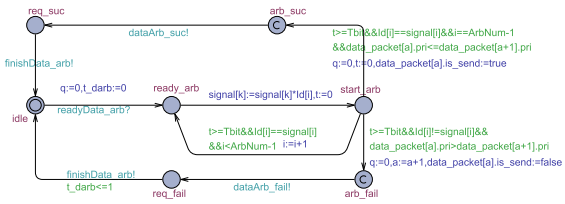

Fig. 8. Arbiter model

6) Gateway model: After arbitration the message enters the gateway. The speed at which the message is sent is declared in the data packet. If its speed is the same as the next message, we infer that no speed change is required; if it is different, the speed of the message needs to be performed. The change of speed is required in 3 time units. The working principle of the gateway is shown in Figure 9:

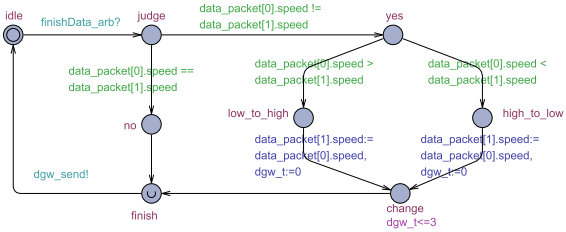

Fig. 9. Gateway model

7) CAN bus model: The CAN bus has two states in the model: idle state and busy state. When the message is delivered, the required number of buses *data_account* is greater than 0, *data_active* will be set to 1 and become busy. After sending the message, data_active is restored to 0. The bus also returns to the idle state. The CAN bus model is shown in Figure 10:

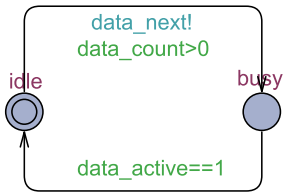

Fig. 10. CAN bus model

V. Formal Description and Verification of Properties

A. Deadlock free

Before verifying other properties, first make sure that no deadlock occurs throughout the communication process.

A[] not deadlock

B. The correctness of the communication system

A feedback is always received after sending message, indicating that the message was successfully transmitted during this time and the actuator performed the corresponding operation.

A<> (Sensor.sending imply Sensor.finish)

C. Arbitration time

UPPAAL expresses time with time units. We set the time to spend a communication cycle *SendPeriod* to 16 time units, then the more time-consuming part of this is the arbitration time and the judgement time of gateway, thus considering the real-time nature of the system, the arbitration time is required to be no more than 3 time units.

A<> t_darb≤3

D. Priority judgment

According to the arbitration mechanism of the CAN bus, high-priority messages are required to be sent earlier than low-priority messages.

A<> data_packet[a].is_send==true imply
data_packet[a+1].is_send==true

E. Gateway judgment time

Same as the arbitration time, the shift time of the gateway is required to be no more than 3 time units.

A<> dgw_t≤3

F. Anti-collision warning

Before the dangerous collision occurs, an alarm signal is issued in time to remind the user. Currently widely used CPS, such as: UAV is equipped with ultrasonic sensors, when the distance from the physical environment is lower than the minimum safe distance during operation, the sensor transmits data to the controller, which will trigger the alarm system to sound to the user and other forms of early warning. Here we require that the execution process be no more than 5 time units, which ensures that the system is able to handle the sudden situations in a short time of the communication cycle.

A<> Range_sensor==1 imply (AWS_system==1 and
sound_warning==1 and AWS_reflect==true and
actuator.t_execute≤5)

G. Electricity warning

CPS has a monitoring mechanism for its own power. We require that when the power is less than 10

A<> battery≤1 imply (sound_warning==1 and
battery_reflect==true and actuator.t_execute≤3)

TABLE II
VERIFICATION CONCLUSION

Property name	Total time(s)	Peak resident memory usage(KB)	Peak virtual memory usage(KB)	Results
Deadlock free	0.020	8964	29988	Satisfy
Correctness	0.016	8408	28852	Satisfy
Arbitration time	0.001	8968	29992	Satisfy
Priority judgment	0.010	8724	30004	Violate
Gateway judge time	0.020	8976	30016	Satisfy
Anti-collision warning	0.011	8984	29453	Satisfy
Electricity warning	0.015	8425	30454	Satisfy

H. Verification results

The computer environment in which UPPAAL software runs in this article is: 3.60GHz eight-core CPU with a memory environment of 8.00GB. Table II shows the verification results of the above conditions in this environment. The total verification time is shorter and all below the second level; the peak usage of resident memory and virtual memory is also within a reasonable range acceptable. The verification results demonstrate that the CPS communication system based on CAN bus can smoothly transmit data and respond to the external situation in time to meet the real-time requirements. In the above table six of the attributes are satisfied except the priority judgment. The reason is that if there are always other nodes sending messages with higher identifiers at the same time, the current node may never win the arbitration and successfully send its message. A node attempting to send a message may never succeed because it has a lower priority. This is why the attribute failed validation. Here we propose to add dynamic priority judgment, the original CAN bus priority is static priority, that is, it will not change. It may result in never being able to send message. When the number of failed arbitrations has reached, the dynamic priority of the node is raised, which will ensure that low priority messages can be sent after a certain time.

VI. CONCLUSION AND FUTURE WORK

The formal verification of CAN bus is studied in this paper. The model is established by abstracting the communication system of CPS modules. The model checking method can effectively verify the logical correctness and real-time feature of the system. Through the verification of UPPAAL, it can ensure that the communication system sends data in time when CPS encounters an emergency situation, and the system responds correctly. One of the contributions of this paper is modeling the gateway in Can bus to ensure the correctness of the message transmission under the difference rate. The other is modeling the priority of the packets in the arbiter. The experimental results also reveal that the design logic of the model provided by this study is feasible, and the method of verifying the property is correct. The next step we may solve the dynamic input transmission content, and also apply the model checking method to other systems and fields.

REFERENCES

[1] Hongzhuan Zhao, Hang Yue, Tianlong Gu, and Wenyong Li. Cps-based reliability enhancement mechanism for vehicular emergency warning system. *International Journal of Intelligent Transportation Systems Research*, 17(3):232–241, 2019.

[2] Chunjie Yang and Ji Yao. The design of distributed control system based on can bus. In *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, volume 8, pages 3956–3958. IEEE, 2011.

[3] Johji Suzuki and Tomoki Saito. Communication system, gateway device and gateway program, September 18 2003. US Patent App. 10/385,534.

[4] Lakshmanarao Battula and P Vamsikrishna Raja. Power efficient gathering in sensor information systems protocol using k-means clustering algorithm. *International Journal of Science, Engineering and Computer Technology*, 6(4):133, 2016.

[5] Jung-Yup Kim, Ill-Woo Park, Jungho Lee, Min-Su Kim, Baek-Kyu Cho, and Jun-Ho Oh. System design and dynamic walking of humanoid robot khr-2. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1431–1436. IEEE, 2005.

[6] Hafizah Mansor, Konstantinos Markantonakis, and Keith Mayes. Can bus risk analysis revisit. In *IFIP International Workshop on Information Security Theory and Practice*, pages 170–179. Springer, 2014.

[7] Umberto Fugiglando, Emanuele Massaro, Paolo Santi, Sebastiano Milardo, Kacem Abida, Rainer Stahlmann, Florian Netter, and Carlo Ratti. Driving behavior analysis through can bus data in an uncontrolled environment. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):737–748, 2018.

[8] Wen Jin, Xi Chen, and Hui Qun Zhang. Modified worst case response time analysis of can bus consisting of nodes with buffers. In *Applied Mechanics and Materials*, volume 513, pages 3393–3396. Trans Tech Publ, 2014.

[9] Can Pan, Jian Guo, Longfei Zhu, Jianqi Shi, Huibiao Zhu, and Xinyun Zhou. Modeling and verification of can bus with application layer using uppaal. *Electronic Notes in Theoretical Computer Science*, 309:31–49, 2014.

[10] Zhicheng Fu, Chunhui Guo, Shangping Ren, Yu Jiang, and Lui Sha. Modeling and integrating physical environment assumptions in medical cyber-physical system design. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 1619–1622. European Design and Automation Association, 2017.

[11] Yu Jiang, Houbing Song, Rui Wang, Ming Gu, Jiaguang Sun, and Lui Sha. Data-centered runtime verification of wireless medical cyber-physical system. *IEEE transactions on industrial informatics*, 13(4):1900–1909, 2016.

[12] Yu Jiang, Lui Sha, Maryam Rahmaniheris, Binhua Wan, Mohammad Hosseini, Pengliu Tan, and Richard B Berlin. Sepsis patient detection and monitor based on auto-bn. *Journal of medical systems*, 40(4):111, 2016.

[13] Yonit Kesten, Oded Maler, Monica Marcus, Amir Pnueli, and Elad Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256(1-2):93–112, 2001.

[14] Hui-Min Lin and Wen-Hui Zhang. Model checking: Theories, techniques and applications. *Acta Electronica Sinica*, 30(12A):1907–1912, 2002.

[15] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[16] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Advanced Course on Petri Nets*, pages 87–124. Springer, 2003.

[17] Renjun Li, Chu Liu, and Feng Luo. A design for automotive can bus monitoring system. In *2008 IEEE Vehicle Power and Propulsion Conference*, pages 1–5. IEEE, 2008.

[18] Dai Qiang Wang, Shiyou Gao, Yu Qing Chen, Yi Wang, and Qiao Liu. Intelligent control system based on can-bus for car doors and windows. In *2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication*, pages 242–245. IEEE, 2009.

[19] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114, 2002.

[20] Che-Hao Chuang and Ming-Dou Ker. System-level esd protection for automotive electronics by co-design of tvs and can transceiver chips. *IEEE Transactions on Device and Materials Reliability*, 17(3):570–576, 2017.

[21] Huan-Kai Peng and Youn-Long Lin. An optimal warning-zone-length assignment algorithm for real-time and multiple-qos on-chip bus arbitration. *ACM Transactions on Embedded Computing Systems (TECS)*, 9(4):35, 2010.

[22] Elias Bou-Harb. Passive inference of attacks on scada communication protocols. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

[23] Yu Jiang, Han Liu, Houbing Song, Hui Kong, Rui Wang, Yong Guan, and Lui Sha. Safety-assured model-driven design of the multifunction vehicle bus controller. *IEEE Transactions on Intelligent Transportation Systems*, 19(10):3320–3333, 2018.

[24] Mohammad Mahdi Jaghoori, Frank S de Boer, Tom Chothia, and Marjan Sirjani. Schedulability of asynchronous real-time concurrent objects. *The Journal of Logic and Algebraic Programming*, 78(5):402–416, 2009.