Urban Rail Transit (2022) 8(1):45–55
https://doi.org/10.1007/s40864-022-00165-y

http://www.urt.cn/

ORIGINAL RESEARCH PAPERS

Architecture Design and Reliability Evaluation of a Novel Software-Defined Train Control System

Ming Chang[1,2] · **Nan Nan**[3] · **Dongxiu Ou**[3] · **Lei Zhang**[1]

Received: 5 April 2021 / Revised: 12 December 2021 / Accepted: 17 January 2022 / Published online: 17 February 2022

Abstract Communication-based train control (CBTC) has been the prevailing technology of the urban transit signaling system. However, CBTC also faces a few issues to extend and maintain because of its complicated structure. This paper presents a novel urban transit signaling system architecture, software-defined train control (SDTC), which is based on cloud and high-speed wireless communication technology. The core functions of the proposed SDTC, including the onboard controller, are implemented in the cloud platform, with only sensors and input–output (IO) units remaining on the trackside and the train. Because of the scalable framework, the system function can be expanded according to the user's demand, making signaling as a service possible. With warm standby server redundancy, SDTC has better reliability. Compared with the traditional CBTC architecture, the mean time between failures is improved by 39% by calculating typical project parameters by the Markov model based on some assumptions.

Keywords Communication-based train control (CBTC) · Software-defined · Train control · Cloud computing · System architecture · Reliability

✉ Dongxiu Ou
Ou.dongxiu@tongji.edu.cn

Ming Chang
lancechang@outlook.com

Nan Nan
Nannan@casco.com.cn

Lei Zhang
reizhg@tongji.edu.cn

[1] The Key Laboratory of Road and Traffic Engineering, Ministry of Education, School of Transportation Engineering, Tongji University, 4800 Cao' an Rd., Shanghai, China

[2] CASCO Signal Ltd., No.14-15, Lane 299, Wenshui Rd, Shanghai, China

[3] Shanghai Key Laboratory of Rail Infrastructure Durability and System Safety, School of Transportation Engineering, Tongji University, 4800 Cao' an Rd., Shanghai, China

Communicated by Tao Tang.

1 Introduction

Communication-based train control (CBTC) has been the prevailing train control system in urban rail transit. The conventional CBTC mainly consists of the zone controller (ZC), the computer interlock (CI), the automatic train supervision (ATS) on the track side, and the vehicle onboard controller (VOBC) on the train. The CBTC system is mature and reliable, which can provide safe and efficient service for the operation of urban rail transit [1, 2].

It is worth noting that some defects have been exposed with the worldwide application of the CBTC system. For example, the VOBC, ZC, and CI all contain safe logic operations, among which the interfaces are complex, and the subsystems have different hardware boards, which cannot be interchangeable. This will increase the difficulty of installation and maintenance.

Two trends have emerged for optimizing CBTC architectures and reducing complexity. One is to simplify trackside equipment and allow the onboard controller to perform more logical computing functions, i.e., a train-centric or train-to-train (T2T) communication-based architecture. Alstom's T2T CBTC Urbalis Fluence system will be put into operation in Lille [3]. Chen et al. proposed a new train-centric CBTC solution using the information entropy method to compare the structure and function

complexity between the conventional CBTC system and the train-centered system [4]. Liu et al. proposed a train control system based on train-to-train communication (T2T-CBTC) and designed a risk prediction method estimated by a statistical model test [5]. Song et al. reviewed the development of train control systems. The authors put forward a train-centric architecture and analyzed the system availability and performance mainly with respect to communication by Petri Net [6–8]. Wang et al. introduced a new sense-based semipermanent scheduling method for resource allocation to improve the transmission delay of the T2T system, and the method also evaluated the quality of service and how to defend against Sybil attacks [9–11].

Another idea for optimizing CBTC architectures is to virtualize the trackside controllers currently placed at each equipment station and deploy them in the cloud center. This enables the use of commercial off-the-shelf (COTS) servers instead of dedicated industrial control computers, which reduces the costs and simplifies the maintenance operations. Increasing numbers of maintenance and train management systems are going to the cloud for powerful data analysis, intelligent diagnostics, and dispatch management functions. In addition, Siemens' interlock based on the Distributed Smart Safe System (DS3) platform has been in operation in Achau, Austria, since 2020 [12]. DS3 is a safety integrity level 4 (SIL4) product implemented on COTS hardware, demonstrating that it is possible to transfer safety-critical products into the cloud. The Swiss Railways' SmartRail 4.0 project proposes that the European Train Control System (ETCS) interlocking system will be implemented in the cloud and has commissioned ESG Rail to analyze the requirements and the potential routes for the SIL4 data center [13].

Both train-centric and cloud architecture trends are evolving towards simplifying the CBTC architecture and reducing the system complexity. However, the division of system component functions in these schemes is still limited to their physical location. Even in the case of "interlocking in the cloud," the allocation of functions to the components of the train control system is similar to that of conventional architecture.

In fact, with the development of modern wireless communication technologies, it is possible to break the space barrier. As one of the ultra-reliable low-latency communication (uRLLC) scenarios, the train control system has been defined in the 5G Release 16 of 3GPP TR 22.289: the end-to-end transmission delay shall be less than 100 ms and the reliability shall reach 99.9999% [14]. Research on the application of 5G to transportation, such as vehicle-to-everything, vehicle networks, etc., has begun to appear [15–18]. If the train-to-ground wireless communication can reach millisecond delays with extremely high reliability, then there will be no physical separation between train and ground boundaries. That is, not only can the interlocking or ZC be put into the cloud, but the logical computing units of the onboard controller can also be put into the cloud, resulting in greater flexibility and reliability.

In this paper, a new train control system architecture for urban rail transit is proposed, the so-called Software-defined Train Control (SDTC) system, and the reliability indices of the architecture are evaluated. The contributions of the SDTC are as follows:

(1) The SDTC's logic computing functions are deployed in the cloud, and each subsystem shares a common safety and redundancy architecture to solve maintenance problems for the hardware of the traditional system.
(2) Considering resource management as the core concept, SDTC regards physical devices as resources and divides entity, control, and applications into vertical layers, which simplifies the system. Also, it is convenient for migration, expansion, and management, making signaling as a service possible.
(3) Due to the low delay and high-reliability communication technology, there are only sensors and actuators on the train, and the control model is in the cloud, which makes the upgrade and expansion easier.
(4) The reliability index of SDTC is improved due to the use of the Markov model to model and quantitatively evaluate the conventional CBTC and SDTC.

Section 1 in this paper presents an introduction and literature review, Sect. 2 presents the architecture of SDTC, and Sect. 3 describes the reliability model used to evaluate the CBTC and SDTC system and discusses the quantified results. Conclusions are made in Sect. 4.

2 SDTC Architecture

2.1 CBTC System Structure

The arrangement of traditional CBTC equipment is shown in Fig. 1, which does not consider backup equipment, such as the track circuit or Eurobalises. The overall CBTC system architecture is divided into three levels: a control center, a station, and a terminal.

In the control center, there are central ATS devices for dispatching functions. A line controller, such as a data storage unit (DSU), manages data versions and temporary speed limits. The maintenance support system (MSS) will be set up in a maintenance center for the status monitoring and maintenance functions.

Normally, among three to five stations, there is an equipped station (usually with switches) that is deployed

with CI, ZC, and local ATS devices. The other stations are unequipped stations, linked by a cable to the equipped one. The CI can be divided into a logic calculation unit and an input–output (IO) unit called Object Controller (OC). The OC connects to wayside devices, such as switches, signals, platform doors, etc. The logic computing unit of the interlock runs on the safety hardware platform, integrated by 2 out of 2 (2oo2) or 3 out of 2 (3oo2) architectures to ensure safety and reliability. Local ATS is used to take over the equipment control of the equipped station area in the event of disconnection from the center. The ZC is used to obtain the track state from the CI and the running state from the VOBC and to calculate the moving authorization of the train.

On the train, as shown in Fig. 2, the VOBC is deployed at the head and tail cabs to form a hot standby redundancy. For each VOBC device, there are three main parts: the speed measurement and localization unit, the IO unit as the interface with rolling stock, and the core unit for ATP and ATO logic operation. Moreover, the ID plug can identify the train where the VOBC is installed.

2.2 Architecture of SDTC

One way to enhance system scalability is through software-defined functions on general-purpose hardware devices, from the earliest software-defined networks, storage, data center, and cloud to the industrial field [19], such as software-defined power grids [20], vehicular [21], and the internet of things [22]. It is time for software to define everything. The core of the software-defined concept is to disentangle the applications from their specific hardware and to provide a common interface between them. As for the signaling system, whether ZC, CI, or VOBC, if functions can be separated from their specific hardware, it can minimize the variety of devices while preserving system flexibility.

This paper proposes a two-layer structure train control system, where the logical operation layer is located in the cloud, and only the IO and sensors are kept on the rail and train sides. Figure 3 shows the schematic diagram of the SDTC, which consists of the safety-related cloud parts, including the controller in cloud (CiC) for each train, the line resource manager (LRM), a train registration and allocation controller (TRAC), a multiple IO unit installed on the train, and a wayside IO unit installed on the side of the track. Between the cloud and the train, there are low delay, high reliability, and high-speed wireless communication. The cloud is deployed on the control center, and the backup servers can be placed in the same center or the backup center, which connects through a high-speed network and is transparent to the application. Thus, in the event of a major disaster, it is possible to transfer between multiple cloud centers, which minimizes the impact on system operations.

The CiC runs on the SIL4 cloud platform and performs the core logic computing functions of a traditional VOBC. It establishes communication with the Multi-IO module on the corresponding train according to the TRAC assignment, receives real-time speed and location from the train, and sends emergency braking or traction commands to the Multi-IO. The CiC obtains the switch status on the line from the LRM and the position of the ahead train by other

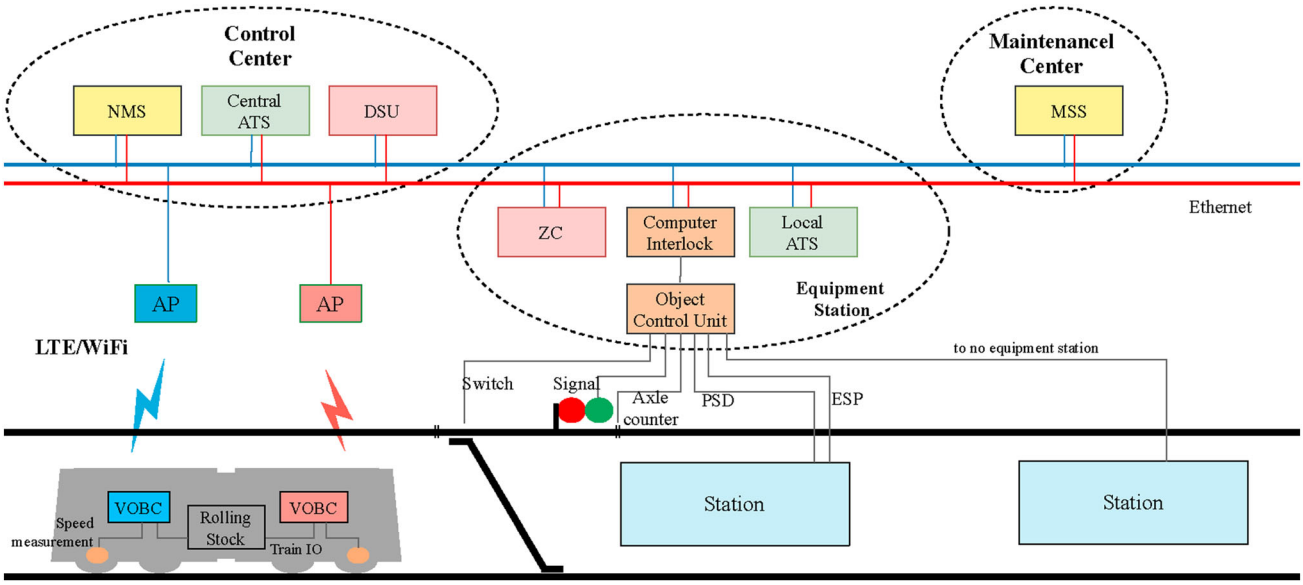

Fig. 1 Architecture of a conventional CBTC system (without the backup system)

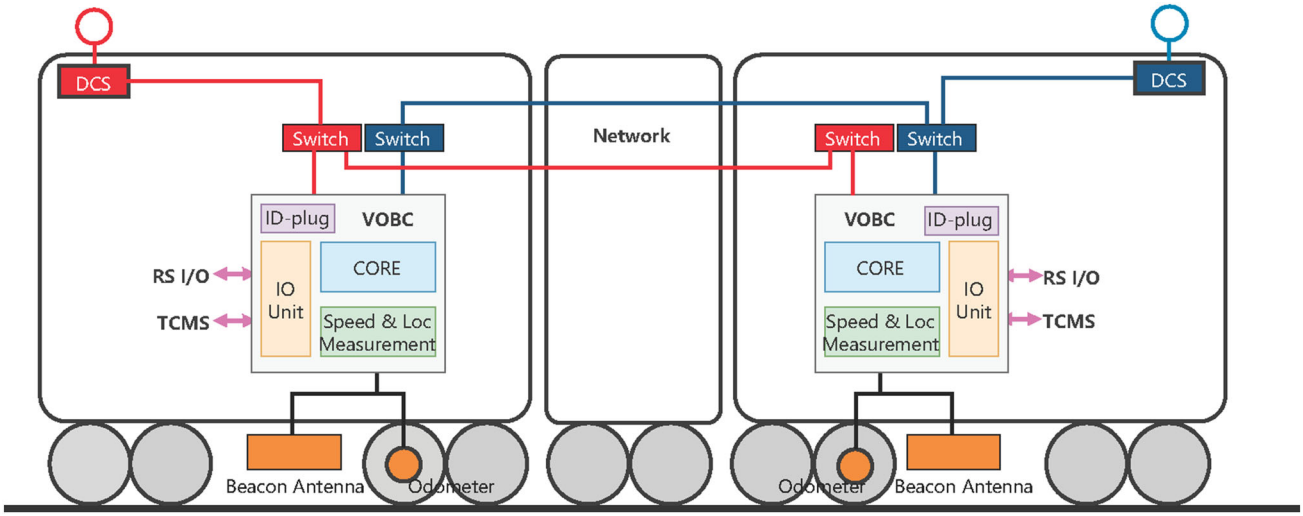

Fig. 2 VOBC architecture on the train

CiCs, and accordingly calculates movement authorizations and door opening and closing commands.

The TRAC assumes two functions: one is to match and manage the Multi-IO and CiC in the cloud, i.e., assigning the corresponding CiC to the new train when it comes into service. This includes informing two CiCs on the same train that they are mutually redundant. The other function is to inspect the working status of the CiC. If one CiC fails,

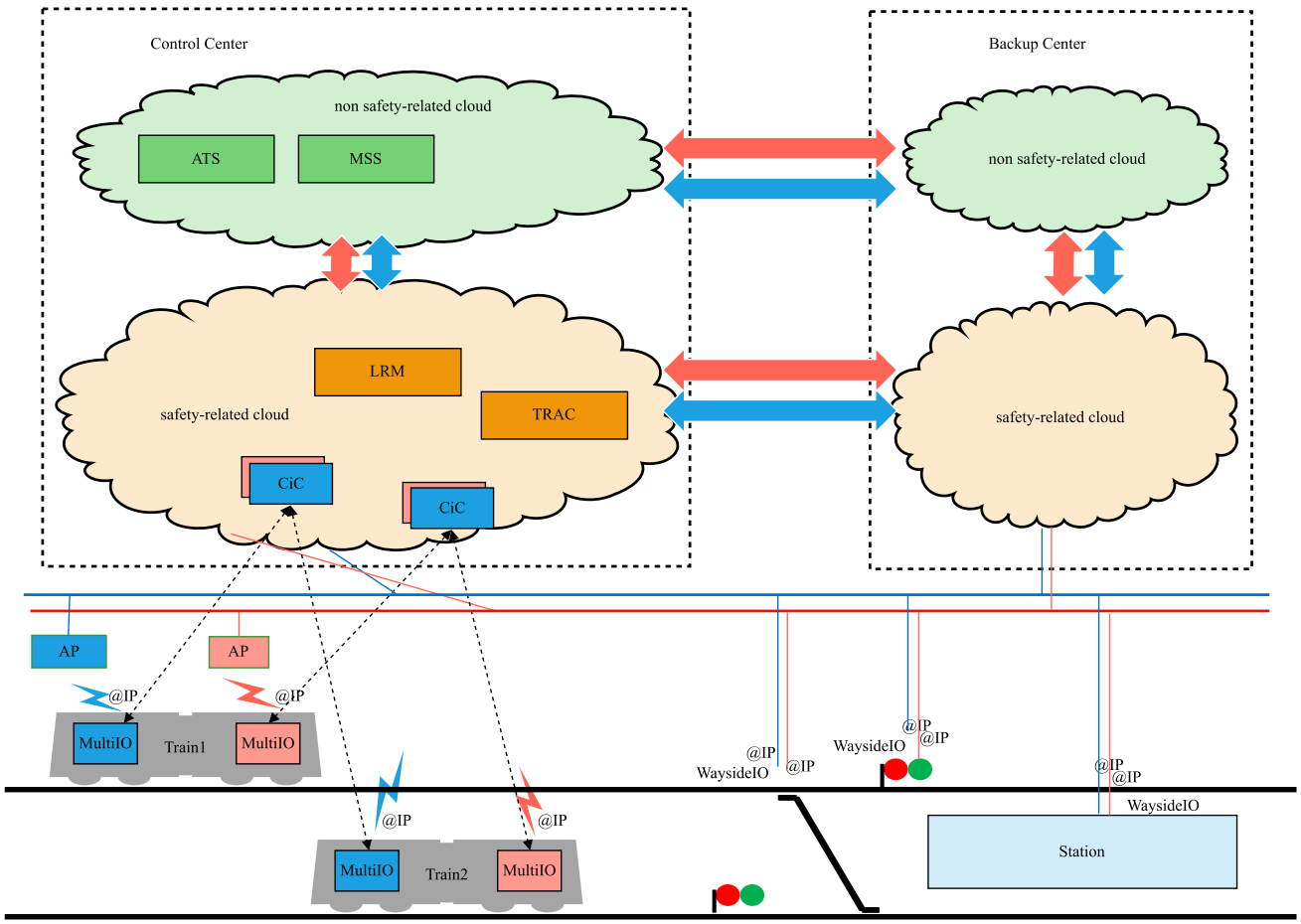

Fig. 3 SDTC architecture

it can request the cloud platform to cut off the failed CiC server and replace it with the standby server, to enhance the availability of the entire system.

The LRM manages and distributes line resources. Through communication with the Wayside-IO, it obtains real-time information about the trackside equipment, such as the position of switches, the platform door status, etc. The LRM also needs to maintain temporary speed limits across the line and performs other features such as data version management for other subsystems.

The Multi-IO, placed on the train, is used to realize the communication and interface with the train and man–machine interface display, and to obtain the information of the speed sensor, Eurobalise antenna, Doppler radar, and other equipment installed beneath the train. In addition, the Multi-IO does not need complex processing ability, but only enables communication with the rolling stock's equipment and sensor status acquisition in short cycles, and sends it to the CiC in the cloud. It also receives the command from the CiC and sends it to the rolling stock or man–machine interface for display. Perhaps with the integration of signaling devices with rolling stock TCMS, the Multi-IO will no longer be required, and train driving can be done by the CiC communicating directly with the TCMS.

The Wayside-IO, a device installed on the trackside, is used to communicate with basic signaling devices such as switch controllers, signals, platform doors, etc. It obtains the status of these devices and sends them back to the LRM, or it actuates the switch action, flashes the signal, or opens and closes the platform doors.

As the core logic is implemented in the cloud platform, the SDTC system provides more flexible and easier methods to maintain features than the traditional CBTC architecture. The resources of the cloud can depend on the customer's demand; they do not need excessive investment in the early stages, and can also increase along with the system expansion.

2.3 Safety and Redundancy Mechanisms

For the train control system, safety and reliability are the most important characteristics, which must be verified by hardware redundancy to avoid the dangerous results of the failures. The multicore processor is a virtualization platform that enables a SIL4 level safety vote to avoid random hardware failures [23]. Figure 4 shows a safety train control cloud platform based on multicore servers. It is used to implement safety computing on commercial servers, avoid random errors through multicore voting, and support the implementation of SIL4-level functional applications. Software encoding can be used to reduce the undetectable rate of random faults on COTS devices [24]. Furthermore, to enhance the diversity between the channels, different software encoding can be applied in different central processing unit (CPU) cores of the 2oo2 architecture or using different operating systems or compilers, thus further reducing the probability of common cause failure in the system.

The examples in Fig. 4 are four multicore servers (#1, #2, #x, and #y) with multiple cores per server. With a hypervisor, the cores are separated and mapped to separate CPUs to run their respective virtual machine applications. On this virtual machine, a real-time operating system and corresponding train control application (such as CiC, LRM, or TRAC) are run to achieve the corresponding functions. Between the server, and through the high-speed network connection, 2oo2 votes to compare might be realized.

For example, for the CiC controller in Fig. 4 (between server #1 and #2, and between server #x and #y) and the virtual machine corresponding to the core of the respective server, it composes a pair of 2oo2 architecture controllers to realize the safety operation of SIL4-level function. Besides, two pairs of servers, running onboard controllers, can be associated with the same train and form a double 2oo2 redundant architecture. If one server fails, the CiC application running on the other server can control the train and avoid emergency braking or other conditions that could affect train operations.

Similar to the original CBTC, a hot standby in the SDTC is required to maintain uninterrupted service. To increase availability, the SDTC can set up additional warm servers in the cloud, which can quickly replace crashed working devices, thus restoring the hot standby redundancy [25, 26]. If this mechanism was used in the traditional architecture, extra warm standby hardware had to be deployed for every specific subsystem, thus leading to unaffordable costs. In the cloud, however, the same warm standby server can be used for all the working servers, including the CiC, the LRM, or the TRAC, simply by migrating the virtual machine to another server. In this way, the overall system reliability is strengthened by adding a few servers.

Figure 5 presents an example of how the redundancy mechanism works in the SDTC. The TRAC runs on the cloud platform and traces the correspondence between the CiC application and the Multi-IO, as shown in Fig. 5. When the TRAC receives registration information from the Multi-IO, it determines whether there is already a CiC corresponding to it. If not, it assigns and creates a new CiC and informs the ID to bind it.

TRAC communicates with each CiC periodically. When it detects a CiC crash (CiCx-1 on server #x in Fig. 5, for example), the TRAC isolates the CiC's server #x and wakes up the warm standby server #z to replace the failed one. During the cut-and-replace process, Train 1 and Train 2 are controlled by the CiC that is solely deployed on

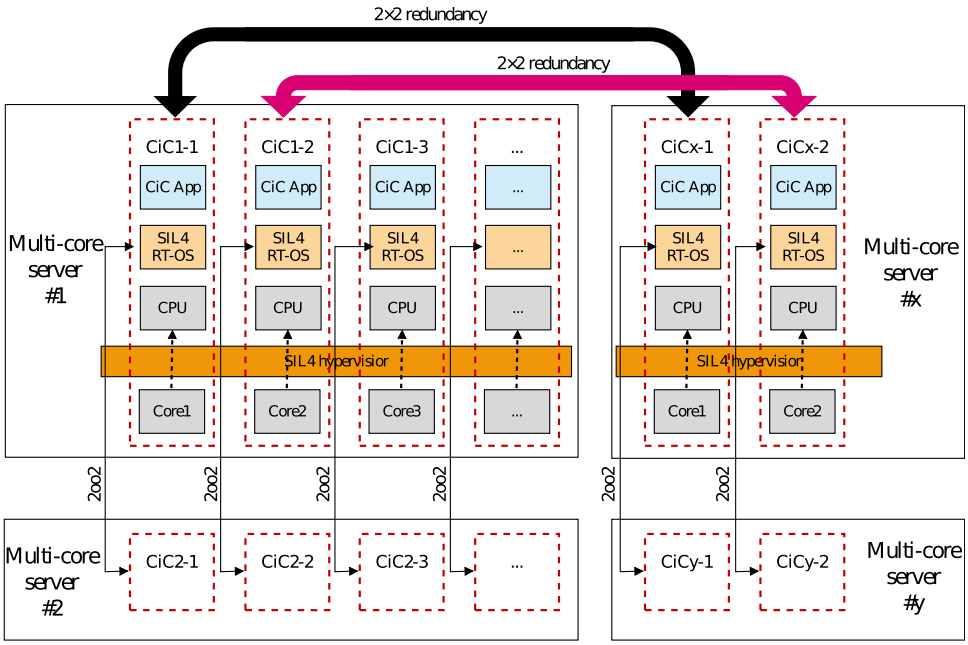

Fig. 4 Safety and redundancy of the servers

servers #1 and #2. After all the applications in the #x server relocated to the server #z, Train 1 and 2's CiC is recovered to double 2oo2 redundant architecture. It should be noticed that server #z is not a specific CiC backup; however, it can be used to replace any failed server. In the following section, the reliability index of SDTC is calculated and compared with the classic CBTC framework.

3 Reliability Evaluation

3.1 RAM (Reliability, Availability and Maintainability) Model of CBTC

Mean time between failures (MTBF) is the index to measure the system reliability. The reciprocal of MTBF, expressed in λ, is the failure rate of the system, as shown in Eq. 1. According to the failure rate of the basic components and the structural model of the system, the failure rate of the subsystem and the entire system can be obtained gradually through the fault tree analysis and Markov state transition matrix [27].

$$\lambda = \frac{1}{\text{MTBF}} \tag{1}$$

Here the key components that make up a CBTC system are identified, which are ATS, VOBC, ZC, DSU, CI, OC, and DCS, and the failure of any one of those key components will cause the system to fail. The MSS is excluded because it does not impact regular operations. The reliability block diagram of a CBTC system can therefore be described as a serial architecture, as shown in Fig. 6a.

For a system consisting of N subsystems in a serial structure, the failure rate λ_{system} can be calculated according to Eq. (2), where $\lambda_{\text{sub\_}i}$ indicates the failure rate of the subsystem i, $(1 - \lambda_{\text{sub\_}i})$ means the probability that the subsystem works properly, so the product of all subsystem $(1 - \lambda_{\text{sub\_}i})$ indicates that the entire system works normally. When the failure rate of each subsystem is very low, the failure rate of the entire system is about equal to the sum of the failure rates of each subsystem.

$$\lambda_{\text{system}} = 1 - \prod_1^N (1 - \lambda_{\text{sub\_}i}) \approx \sum_1^N \lambda_{\text{sub\_}i} \tag{2}$$

Referring to IEEE 1474.1 specifications [1], the availability index A of the CBTC system is the probability that the system is capable of operating its intended function at a random point in time, as determined by the system MTBF and the mean time to repair (MTTR), as shown in Eq. (3).

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \tag{3}$$

For the serial architecture, the MTTR of the entire system is the sum of the product of each subsystem's $\text{MTTR}_{\text{sub\_}i}$ and its failure rate $\lambda_{\text{sub\_}i}$, divided by the failure rate of the system λ_{system}, as shown in Eq. (4).

$$\text{MTTR}_{\text{system}} = \sum_i (\text{MTTR}_{\text{sub\_}i} \times \lambda_{\text{sub\_}i}) / \lambda_{\text{system}} \tag{4}$$

Considering the VOBC shown in Fig. 2 as an example, a Markov model of the redundant architecture is obtained and as shown in Fig. 7. Each train has two sets of VOBC equipment that are redundant with each other. For S0, and under usual circumstances, both ends of the VOBC are working normally, one of which is hosted to control the train, and the other is a backup. Both S1 and S2 are

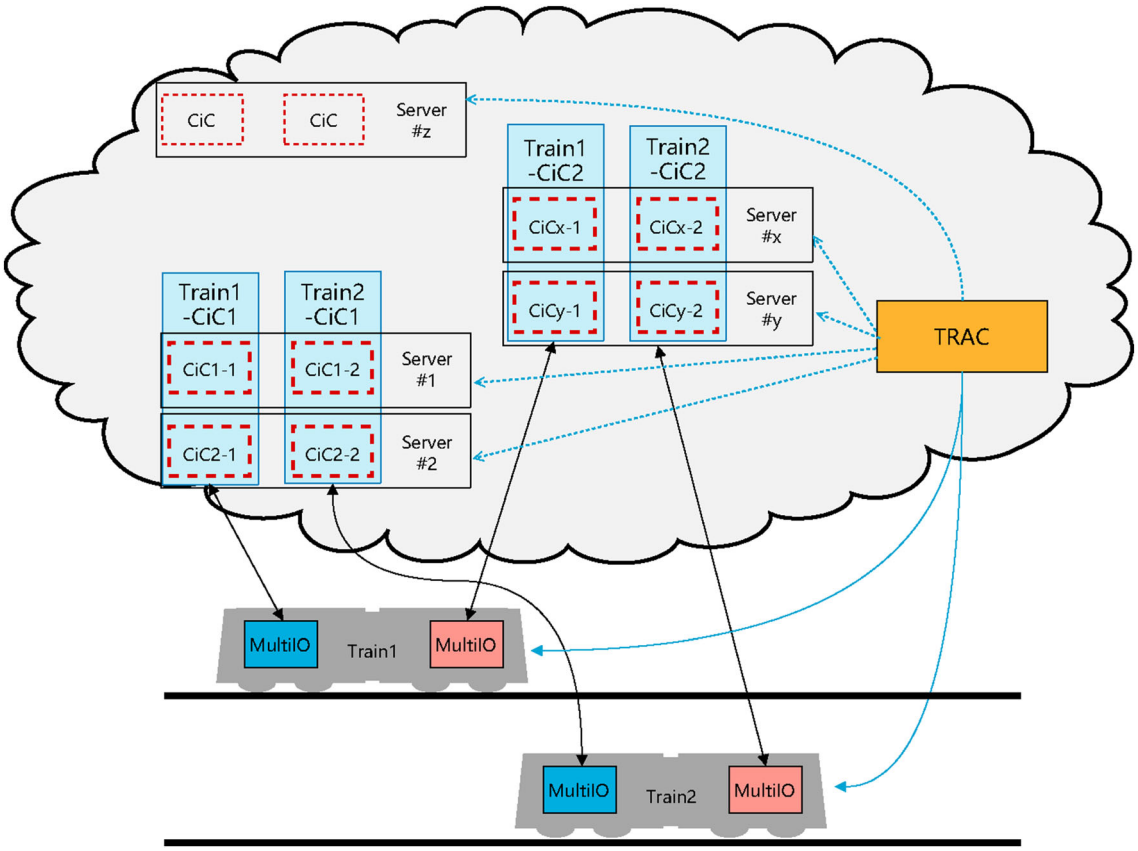

Fig. 5 A scenario of TRAC

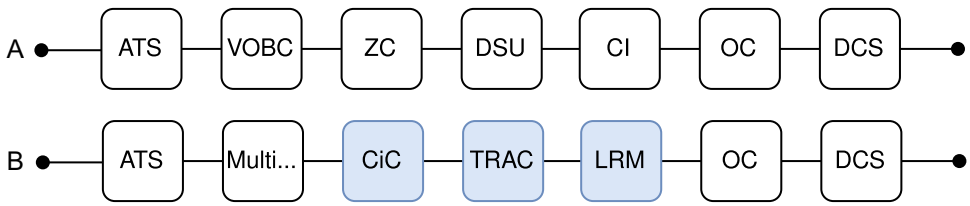

Fig. 6 Reliability block diagram of **a** CBTC and **b** SDTC

degraded cases, with one VOBC out of order. The difference is that S1 is an undetected failure, while S2 is detectable. In both cases, however, the system still has an operating VOBC, so that it is in an available state. S3 is a failed state; that is, both ends of the VOBC are unavailable. Due to the cost, it is not feasible to design different hardware for two VOBCs on the same train; thus, there is a common cause fault in the system, which may lead to dual devices crashing together. Besides, when the failure is detected, or the system is unavailable, repairs should be done as soon as possible. After rebooting, or replacing the failed board, the system resumes to the normal state S0.

According to Fig. 7, the state transition matrix of the system can be obtained as shown in Eq. (5).

$$P = \begin{bmatrix} 1-2\lambda(1-\beta)(1-\alpha)-2\lambda(1-\beta)\alpha-\lambda\beta & 2\lambda(1-\beta)(1-\alpha) & 2\lambda(1-\beta)\alpha & \lambda\beta \\ 0 & 1-\lambda & 0 & \lambda \\ \mu & 0 & 1-\mu-\lambda & \lambda \\ \mu & 0 & 0 & 1-\mu \end{bmatrix} \tag{5}$$

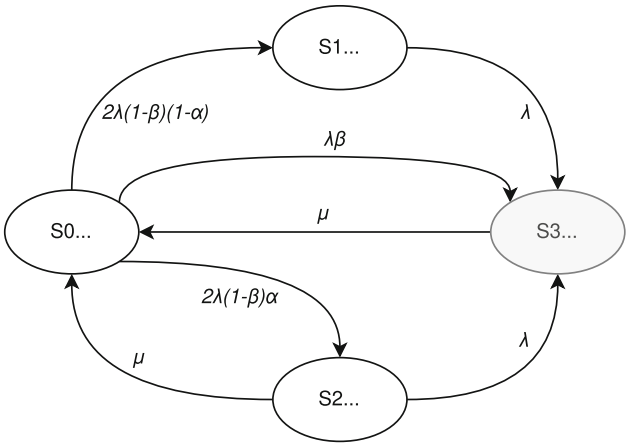

Fig. 7 VOBC Markov model

in which λ is the failure rate of the device, α is the detection rate, β is the common cause failure factor, and μ is the restore rate, reflecting the maintenance rate of the system. Therefore, according to Eq. (6), starting from the initial state S0 = [1,0,0,0], the probability of the system being in each state at any time can be calculated. Along with increasing time, the probability of being in each state will converge on a limited value.

$$[S0_{t+\Delta t}, S1_{t+\Delta t}, S2_{t+\Delta t}, S3_{t+\Delta t}] = [S0_t, S1_t, S2_t, S3_t] \cdot P \quad (6)$$

$$S0 + S1 + S2 + S3 = 1 \quad (7)$$

$$A_{\mathrm{VOBC}} = S0 + S1 + S2 \quad (8)$$

As shown in Eq. (7), the sum of the probabilities of each state is equal to 1, and the availability in Eq. (8) of the system is derived because all states except state 3 are available. According to Eq. (3), Eq. (9) can be deduced for the MTBF of the system.

$$\mathrm{MTBF} = \frac{\mathrm{MTTR} \cdot A}{1 - A} \quad (9)$$

Therefore, the reliability indexes of each subsystem in the CBTC system can be calculated in a sequence, and the reliability of the entire CBTC system can be obtained using the serial structure shown in Fig. 6a and Eqs. 1, 2, 3, 4, and 5.

3.2 SDTC Reliability

For the SDTC architecture shown in Fig. 3, and the RAM model of SDTC shown in Fig. 6b, it is clear that ATS, Multi-IO, CiC, TRAC, LRM, OC, and DCS are key components. Therefore, we can calculate the SDTC index similarly to the CBTC system.

In the SDTC system, TRAC, CiC, and LRM are all deployed in the safety-related cloud, with the same redundancy mechanism shown in Fig. 5. Compared with the classic hardware architecture, the warm standby server in the cloud platform is not constrained to a specific device. Thus, the platform and the data are separated. As long as the specific configuration is loaded, it can act as that device, which is one of the advantages of SDTC systems.

The Markov model can also be used to evaluate the reliability of servers in the cloud [28]. The following assumptions were made: (1) the fault server is only repaired after being isolated; (2) the working server will not recover directly after failure, but will be replaced by the warm standby server; (3) there are common cause faults between redundant working servers; (4) whether or not the fault is detectable does not affect the overall availability of the system, so this model does not consider the detection rate; and (5) there are several warm standby servers represented as W0, W1, etc.

Therefore, the running state of a subsystem in the SDTC and its transition diagram is shown in Fig. 8. S0 is a natural state, comprising two redundant hot standby working servers and a warm standby server W0. When a working server crashes, the system transfers into S1 with a single working server and an inactive warm standby W0. Normally, warm standby W0 will be activated into S3, thus restoring one out of two structures, while the second backup server W1 becomes a warm standby. Otherwise, if the rest of the working system fails when the W0 has not yet been activated, it goes from S1 to failure state S2, at which point the system becomes unavailable. Similarly, if while in the S3 state there is a working server failure, the system will be put into the degraded state S4. When the isolated failure server recovers, the system returns to a normal state S0. Note that if the working one fails at the state S4, it will go to the fault state S5 before warm standby W1 is activated.

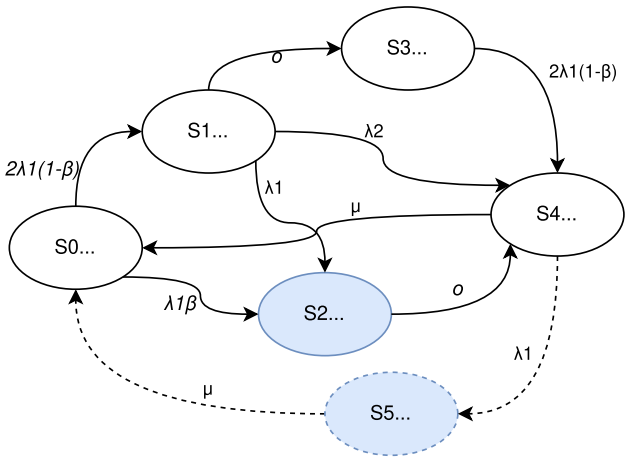

Fig. 8 Markov model for the CiC subsystem with two warm standby servers

State S0 is the same as S3, S1 is the same as S4, and S2 is the same as S5, but only with a different server W0 or W1 on standby. If the system has other warm standby servers, it will have a more identical status. However, because of the rapid activation time, unless a common mode failure occurs, the warm standby system switches in time to resume redundancy before another server fails. Therefore, no further states need to be considered.

The transition matrix for the Markov model shown in Fig. 8 is shown in Eq. (10).

$$\mathrm{P} = \begin{bmatrix} 1-2\lambda_1(1-\beta)-\lambda_1\beta & 2\lambda_1(1-\beta) & \lambda_1\beta & 0 & 0 & 0 \\ 0 & 1-\lambda_1-o-\lambda_2 & \lambda_1 & o & \lambda_2 & 0 \\ 0 & 0 & 1-o & 0 & o & 0 \\ 0 & 0 & 0 & 1-2\lambda_1(1-\beta) & 2\lambda_1(1-\beta) & 0 \\ \mu & 0 & 0 & 0 & 1-\lambda_1 & \lambda_1 \\ \mu & 0 & 0 & 0 & 0 & 1-\mu \end{bmatrix} \tag{10}$$

in which λ_1 and λ_2 are the failure rate of the server in the working state and the warm standby state, respectively, β is a common cause failure factor, μ is the restore rate, and o is the warm standby activation rate.

In this model, the system is unusable when it is in states 2 and 5, whereas it is available in the other states. Thus, the availability of the CiC subsystem for a train is shown in Eq. (11).

$$A_{\mathrm{CiC}} = S0 + S1 + S3 + S4 \tag{11}$$

3.3 System Comparison

To assess the reliability between the conventional CBTC and the new SDTC architecture, an average project is assumed. Thirty trains are running on this line, which is implemented on the CBTC system, with 30 pairs of VOBCs, three ZCs, six CIs, 16 OCs, one ATS, and one DCS. For the SDTC architecture, 30 pairs of CiCs, one LRM, and one TRAC are in the cloud, 30 pairs of Multi-IO on the trains, 16 Wayside-IO outside, and the same one ATS and one DCS system.

For the CBTC system, the parameters of a specific type of CBTC produced by CASCO are adopted, as shown in Table 1. The MTBF value of a single device is calculated in a sequence according to Eqs. 5, 6, 7, 8, and 9, and the MTTR is modeled on analysis. For the VOBC, it is routinely rebooted locally by the driver or remotely by the operator after the train arrives at the terminal station. Here, the running time is presumed to be 1 h. For other subsystems, because their failure will lead to a large range of train emergency brakes, they must be repaired immediately. Considering 0.5 h as MTTR, after obtaining the RAM value of each device of every subsystem, Eqs. 1, 2, 3, and 4 can be used to calculate the overall index of the number of N and the index of the entire CBTC system.

For SDTC systems, the calculation was performed depending on the following principles: (1) ATS and DCS are outside the focus of this study, so the same value of CBTC was used; (2) because of the similar function, the number and the failure rate of Wayside-IO are the same as OC; and (3) Multi-IO is the IO and speed and location measurement, which is two out of three major modules of VOBC in Fig. 2; thus, its failure rate is set as 2/3 of VOBC. Meanwhile, the CiC, LRM, and TRAC structures in the cloud are the same.

With the continuous improvements in technology, the reliability of commercial servers is also improving. LENOVO shows that there are 150,000 hours of MTBF-certified servers [29]. Here, 3 years (i.e., 25,623 hours) was taken as the server's MTBF. According to the Markov model shown in Fig. 8, the failure rate of the server is λ_1 = 3.90274e−05. The failure rate of a warm standby server is less than that of a working server, assuming $\lambda_2 = 0.5 \times \lambda_1$. If the common cause coefficient β is 0.01 and the activation rate time of the warm standby system is 0.01 h (i.e., o = 100), for the MTTR, and due to the faulty server rapidly replacing and restarting in the cloud, it was set to be 0.01 h. Then, according to Eqs. (10) and (11), the RAM of each system can be calculated. Finally, the overall SDTC reliability value is computed as shown in Table 2.

Comparing Table 2 with Table 1, it is clear that the reliability of the system has been improved. In general, the MTBF of SDTC is 8870.56 h, which is about 139% of the MTBF of 6398.14 h in the conventional system, which represents a significant improvement. In the MTTR index, the 0.86 h in the SDTC have decreased by 10% as compared with 0.95 h in the CBTC. However, SDTC is only slightly better than CBTC in availability. Because the existing CBTC provides enough redundant design, a single device failure would not affect the normal operation of the system.

When looking at individual subsystems one by one, for the LRM and TRAC with only one on the line, their

Table 1 RAM for CBTC architecture

Subsystem	Number	MTBF	MTTR	λ_N	MTBF_*N*	MTTR_*N*	*A_N*
ATS	1	2.7778E+05	0.5	3.6000E−06	2.7778E+05	0.5	99.999820%
VOBC	30	2.1252E+05	1	1.4116E−04	7.0840E+03	1	99.985886%
ZC	3	2.5432E+06	0.5	1.1796E−06	8.4774E+05	0.5	99.999941%
DSU	1	2.5432E+06	0.5	3.9320E−07	2.5432E+06	0.5	99.999980%
CI	6	8.0801E+05	0.5	7.4256E−06	1.3467E+05	0.5	99.999629%
OC	16	8.8010E+06	0.5	1.8180E−06	5.5006E+05	0.5	99.999909%
DCS	1	1.3954E+06	0.5	7.1665E−07	1.3954E+06	0.5	99.999964%
CBTC				1.5630E−04	6398.14	0.95	99.985129%

Table 2 RAM for the SDTC architecture

Subsystem	Number	MTBF	MTTR	λ_N	MTBF_N	MTTR_N	A_N
ATS	1	2.7778E+05	0.5	3.6000E−06	2.7778E+05	0.5	99.999820%
Multi-IO	30	3.1878E+05	1	9.4108E−05	1.0626E+04	1	99.990590%
CiC	30	2.5621E+06	0.01	1.1709E−05	8.5404E+04	0.01	99.999988%
LRM	1	2.5621E+06	0.01	3.9030E−07	2.5621E+06	0.01	≈100.00%
TRAC	1	2.5621E+06	0.01	3.9030E−07	2.5621E+06	0.01	≈100.00%
Wayside-IO	16	8.8010E+06	0.5	1.8180E−06	5.5006E+05	0.5	99.999909%
DCS	1	1.3954E+06	0.5	7.1665E−07	1.3954E+06	0.5	99.999964%
SDTC				1.1273E−04	8870.56	0.86	99.990271%

availability is close to 100%, indicating that it is almost impossible to fail. To enhance the reliability of the system, it is key to replace the logic operation function of the VOBC with CiC in the cloud. The MTBF (of 30 pairs of CiC) is 85,404 h, which is 12 times as much as that of the VOBC and shows that the reliability of the system is greatly improved by using a warm standby server architecture to enhance redundancy in the cloud. In the traditional CBTC architecture, it is difficult to use warm standby architecture in the VOBC. Since each train is physically isolated, it would require additional costs to add more hardware. In the cloud, on the other hand, applications are separated from hardware, and it is possible to execute with specific data when it is needed, which provides huge advantages.

4 Conclusions

This paper presents a software-defined train control system, which deploys application software in a safe cloud to implement the logical computing functions of CBTC and to remotely control the train via wireless communication. In comparison to "interlocking in the cloud," the SDTC further enhances the flexibility and reliability of the train control system by deploying the core functions of the VOBC in the cloud as well. According to the reliability calculations, and compared with the traditional CBTC structure, the warm standby server redundancy can greatly enhance the reliability index of the entire system.

Indeed, SDTC is still in the conceptual stage and needs more research and testing. There are two main directions for further research: the wireless performance required for remote control of trains, and whether 5G communication can meet the requirements. Approaches to building safe clouds through COTS hardware are another possible area for future research, such as isolating different CPU cores, diversifying software coding, and synchronizing safe clocks, etc.

Acknowledgements This research was funded by the National Key R&D Program of China (Grant Number 2018YFB1201403), and the Research Program of Shanghai Science and Technology Committee (Grant Numbers 18DZ2202600 and 20511106402).

References

1. IEEE 1474.1TM (2004). IEEE standard for communications-based train control (CBTC) performance and functional

requirements. rail transit vehicle interface standards committee of the IEEE vehicular technology society.
2. Diemunsch K, and Rabindran N (2020) Origins and current status of the different communications-based train control products. JRC2020. doi: https://doi.org/10.1115/JRC2020-8020
3. Briginshaw D (2013) Alstrom's simplified CBTC technology to debut in Lille. International Railway Journal, 53(6). https://trid.trb.org/view/1253251
4. Chen T, Wang H, Ning B, Zhang Y, Tang T, Li K (2018) Architecture design of a novel train-centric CBTC system. Int Conf Intell Rail Transp (ICIRT) 2018:1–5.
5. Liu J, Zhang Y, Han J, He J, Sun J, Zhou T (2019) Intelligent hazard-risk prediction model for train control systems. IEEE Trans Intell Transp Syst. https://doi.org/10.1109/TITS.2019.2945333
6. Song H, Schnieder E (2018) Development and evaluation procedure of the train-centric communication-based system. IEEE Trans Veh Technol. https://doi.org/10.1109/tvt.2018.2868881
7. Song H, Schnieder E (2019) Availability and performance analysis of train-to-train data communication system. IEEE Trans Intell Transp Syst 20(7):2786–2795.
8. Song H, Wu W, Dong H, Schnieder E (2018) Propagation and safety analysis of the train-to-train communication system. IET Microw Antennas Propag. https://doi.org/10.1049/iet-map.2018.6074
9. Wang X, Liu L, Tang T, Zhu L (2018) Next generation train-centric communication-based train control system with train-to-train (T2T) communications. Int Conf Intell Rail Transp (ICIRT) 2018:1–5.
10. Wang X, Liu L, Zhu L, Tang T (2019) Joint security and QoS provisioning in train-centric CBTC systems under sybil attacks. IEEE Access. https://doi.org/10.1109/access.2019.2927048
11. Wang X, Liu L, Zhu L, Tang T (2019) Train-centric CBTC meets age of information in train-to-train communications. IEEE Trans Intell Transp Syst. https://doi.org/10.1109/TITS.2019.2936219
12. SIEMENS. (2020). First hardware independent cloud-enabled interlocking in operation. 2020-11-26. https://press.siemens.com/global/en/pressrelease/first-signalling-cloud-operation.
13. ESG. (2018) On Design, Introduction and Operation Of Safety-critical Applications in a Data Center In the Railway System of Schweizerische Bundesbahnen SBB. 2018–06–14. https://www.smartrail40.ch/download/downloads/Safety-critical%20Applications%20in%20Data%20Center%20in%20the%20Railway%20System%20SBB.pdf
14. 3GPP. TR 22.289 (2019). Technical specification group services and system aspects: mobile communication system for railways(Release 16).
15. Ashraf SA, Blasco R, Do H, Fodor G, Zhang C, Sun W (2020) Supporting vehicle-to-everything services by 5G New radio release-16 systems. IEEE Commun Stand Magazine 4(1):26–32. https://doi.org/10.1109/MCOMSTD.001.1900047
16. Ge X (2019) Ultra-reliable low-latency communications in autonomous vehicular networks. IEEE Trans Veh Technol 68(5):5005–5016.
17. Sadio O, Ngom I, Lishou C (2020) Design and prototyping of a software defined vehicular networking. IEEE Trans Veh Technol 69(1):842–850. https://doi.org/10.1109/tvt.2019.2950426
18. Su Y, Lu X, Huang L, Du X, Guizani M (2019) A novel DCT-based compression scheme for 5g vehicular networks. IEEE Trans Veh Technol 68(11):10872–10881. https://doi.org/10.1109/tvt.2019.2939619
19. Jararweh Y, Al-Ayyoub M, Darabseh A, Benkhelifa E, Vouk M, Rindos A (2016) Software defined cloud: Survey, system and evaluation. Futur Gener Comput Syst 58:56–74. https://doi.org/10.1016/j.future.2015.10.015
20. Cao X, Huang H, Wang X et al (2016) Software defined grid: concept, architecture and samples. Autom Electric Power Syst 40(6):1–9
21. Han S, Cao D, Li L, Li L, Li SE, Zheng N-N, Wang F-Y (2019) From software-defined vehicles to self-driving vehicles: a report on CPSS-based parallel driving. IEEE Intell Transp Syst Mag 11(1):6–14. https://doi.org/10.1109/MITS.2018.28765755
22. Mavromatis A, Colman-Meixner C, Silva AP, Vasilakos X, Nejabati R, Simeonidou D (2020) A Software-defined IoT device management framework for edge and cloud computing. IEEE Internet Things J 7(3):1718–1735. https://doi.org/10.1109/JIOT.2019.2949629
23. Pérez Tijero H, Aldea Rivas M, Medina Ortega D (2017) Multiprocessor platform for partitioned real time systems. Softw Pract Exper 47(1):61–78.
24. Ghadhab M, Kaienburg J, Süßkraut M, Fetzer C (2016) Is Software coded processing an answer to the execution integrity challenge of current and future automotive software-intensive applications? advanced microsystems for automotive applications. Springer, Cham
25. Srinivasa Rao TSS, Gupta UC (2000) Performance modelling of the M/G/1 machine repairman problem with cold-, warm- and hot-standbys. Comput Ind Eng 38(2):251–267.
26. Sousa E, Lins F, Tavares E, Maciel P (2017) Cloud infrastructure planning considering different redundancy mechanisms. Computing 99(9):841–864.
27. Bukowski JV, Goble WM (1995) Using Markov models for safety analysis of programmable electronic systems. ISA Trans 34(2):193–198. https://doi.org/10.1016/0019-0578(95)00008-N
28. Matos R, Dantas J, Araujo J, Trivedi KS, Maciel P (2017) Redundant eucalyptus private clouds: availability modeling and sensitivity analysis. J Grid Comput 15(1):1–22. https://doi.org/10.1007/s10723-016-9381-z
29. LENOVO. (2019). 150,000 hours MTBF certification. https://club.lenovo.com.cn/thread-5421014-1-1.html. 20 January 2019