Distributed Data Backup and Recovery for Software Defined-WAN Controllers

Yi Zhang, Lujie Zhong, Shujie Yang and Gabriel-Miro Muntean

***Abstract*—Software-defined wide area network (SD-WAN) is a new type of network architecture that has developed rapidly in recent years. SD-WAN inherits the centralized control architecture of Software-defined networking (SDN), but supports more diverse access methods and equipment types and covers a wider area. It is also associated with greater uncertainty in the network environment. These characteristics make the fault management of the SD-WAN controller more challenging, so that the existing SDN-based data backup methods cannot adapt to SD-WAN scenarios. This paper proposes an SD-WAN-oriented Distributed Data Backup and Recovery method (DDBR) based on an improved secret sharing algorithm. To deploy this method, we design an online-offline dual backup framework based on the data freshness requirements of the controller. Under this framework, dynamic data of the controller is divided into different shares, and then stored into the storage of switches. When the controller fails, data recovery can be performed on the backup controller quickly, which greatly improves the network availability. The outstanding feature of the proposed DDBR method is that it ensures the integrity and confidentiality of the backup data in an unreliable network environment, even when some of the storage nodes fail. Theoretical analysis and evaluation results on file backup example show that the proposed solution has significant advantages over existing methods in terms of backup data storage size, communication overhead, scalability and backup success rate.**

***Index Terms*—SD-WAN, data backup, distributed storage, Shamir's secret sharing, controller**

I. Introduction

With the rapid development of 5G networks and cloud computing technologies, enterprise businesses are gradually migrating to the cloud. Meanwhile, the demand for teleworking has also entered a stage of rapid growth with the outbreak of COVID-19. Accessing Enterprise resources through public network has become a regular requirement for employees. The wide-area access requirements make the physical perimeter of enterprise networks no longer relevant, bringing great challenges to the management of enterprise networks [1]. As it brings a new approach to the wide area network management, Software Defined Wide Area Network (SD-WAN) has received extensive attention from the Industry.

By deploying an SD-WAN, enterprise network managers can implement fine-grained configuration of network resources, optimize network performance, reduce management complexity, and save maintenance expenses, so the service efficiency of the overall network can be greatly improved [2]. A large number of enterprises have already invested in this emerging field. The consulting company Futuriom predicted that the total market size of SD-WAN will reach $2.85 billion in 2021 and $4.6 billion by 2023 [3]. However, as an application of Software-Defined Networking (SDN) in a wide area network, SD-WAN inherits some vulnerabilities of SDN, for example, the security issues of SDN controllers. On the one hand, the wide-area accessibility of SD-WAN and the more heterogeneous terminal devices make its controller completely exposed to a large number of security threats, such as DDoS attacks, thereby causing its inherent single point of failure more prominent. On the other hand, security incidents such as power outages, fires, and operational errors may also cause the SD-WAN control servers to go down, thereby paralyzing the entire enterprise network. Therefore, providing a data backup method for the SD-WAN controller is a core issue that affects the availability and security of the whole network.

Fault-tolerant methods are important aspects that continue to receive attention in SDN-related research fields, as they directly affect the availability of the network [4]. Fault-tolerant management includes fault detection, location, correction and prevention, etc. It mainly addresses two types of problems: the single-point failure of the controller, and link failures. The recovery of link failure is involved in the routing strategy design of the control plane and has nothing to do with data backup and recovery. It also has relatively little impact on network performance [5]. In a WAN scenario, some scholars have also carried out related research [6]–[8]. These work mostly focus on quickly recalculating the forwarding rules after link failures without causing network congestion, and cannot be used to solve the single point of failure of the controller. In a SD-WAN context, the recovery after a single point failure of the controller requires the backup of the entire control plane, and the impact on the network performance is very significant. This is the main problem studied in this work. Different from the traditional SDN control plane data backup and recovery method, the data backup process of the SD-WAN control plane may involve frequent remote backup and remote data synchronization, which has higher requirements for the design of backup schemes and data synchronization mechanisms. Besides, considering that the backup data may also be damaged or disabled, the persistence of the backup method should also be considered. However, most of the existing studies pay no attention to these aspects, but focus on a classic SDN context [9]–[11].

To fill this gap, in this paper we propose a Distributed

Y. Zhang and S. Yang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. (e-mail: pili@bupt.edu.cn, sjyang@bupt.edu.cn)

L. Zhong is the corresponding author. She is with the Information Engineering College, Capital Normal University, Beijing 100048, China. (e-mail: zhonglj@cnu.edu.cn).

G. Muntean is with School of Electronic Engineering, Dublin City University, Glasnevin, Dublin, Ireland. (e-mail: gabriel.muntean@dcu.ie)

Data Backup and Recovery method (DDBR) for the SD-WAN architecture based on a secret sharing scheme. The DDBR method utilizes the storage of the switches and performs an online-offline dual backup of the control plane data. The advantage of the proposed method is twofold. On the one hand, DDBR ensures that the backup controller can be quickly restored by extracting backup data from the switches when the main controller is disabled. On the other hand, it ensures the security of the backup data. Even though an external intruder obtains the backup data of multiple switches, it cannot obtain any control plane information. The main contributions of this paper are as follows.

- We propose DDBR, a distributed data backup and recovery method based on an improved Shamir's Secret Sharing (SSS) scheme, to solve the problem of distributed data backup in SD-WAN.
- We design an online-offline dual backup method based on SD-WAN architecture, which significantly reduces the communication overhead and storage cost of data backup while maintaining the freshness of the backup data.
- We create a comparative file backup instance to verify the effectiveness and efficiency of the proposed method.

This paper is a summary of our in-depth research based on our previous work which will be presented in IEEE Globecom 2021 [1]. The rest of this paper is organized as follows. We introduce the related works in Section II. In Section III, we propose the data backup method. The dual backup scheme is presented in Section IV. We analyze the system performance theoretically in Section V. Performance evaluations are included in Section VI and Section VII concludes the paper.

II. Related works

A. Shamir's Secret Sharing Method in SD-WAN Context

In cryptography, secret sharing refers to an information protection mechanism in which a set of authorized subsets of participants work together to recover secrets, while also ensuring protection from enemy attacks. The principle of secret sharing is to divide a secret into several shares. These shares are distributed to different users. Only when a specific subset of users provide their own shares together can the initial secret be reconstructed. Secret sharing can effectively prevent attacks by enemies outside the system and betrayal by users inside the system. As a pioneer work, SSS method, proposed in 1979 [12], is a kind of (t, n) threshold scheme. SSS divides the secret into n shares and distributes them to different users. When any t shares are known, it is easy to calculate the secret, and when any less than t shares are known, the secret cannot be obtained. SSS has many useful properties including being secure, minimal, extensible, dynamic and flexible.

In recent years, researchers have carried out some research on the application of SSS to communication networks and distributed storage systems [13], [14], even in distributed and multi-user scenarios [15]. However, these works are limited to theoretical analysis. Empirical research in application fields are still required. Therefore, how to apply SSS in actual communication scenarios is still an open issue.

Applying SSS method in an SD-WAN scenario mainly faces the following problems: 1) SSS is a multi-node sharing method of one secret. The backup of SD-WAN control plane should backup many data files, which is a backup task of many secrets on multiple nodes. 2) SD-WAN is a mixed network environment with wired and wireless channels, the communication links are sometimes unreliable. Network congestion, equipment failure, or malicious hijacking can all cause backup failures, and then data recovery cannot be achieved. 3) SSS requires each device to keep a complete backup of the secret. Distributed backup of dynamic SD-WAN controller may require frequent data transmissions, which consumes lots of communication resources. So a solution to address these issues is still in need.

B. Fault Management Methods in SDN

There are many works which focus on the fault management problem including data backup and recovery [16], [17]. Sasaki et al. proposed a rollback architecture which periodically reverts the process of an OpenFlow switch to its pristine state after handling a flow to protect the data plane of SDN [9]. In [18] and [19], the authors proposed a control plane fault-tolerant backup architecture in SDN from different perspectives respectively.

In a WAN scenario, by pre-calculating a new traffic engineering solution and installing a backup tunnel, Zheng et al. [7] came up with a new traffic engineering solution which ensures that when the link failure occurs, the switch can redirect the traffic to the backup tunnel, so as to achieve the rapid recovery of the data plane function, and avoid the common transmission congestion in the recovery process. In [20], the authors paid special attention to the consistency problem in the SDN control plane replication process. They focused on optimizing the processing delay, and proposed a fast and consistent controller replication mode. Aiming at the service quality requirements in network multimedia transmission services, Basu et al. proposed a fault-tolerant backup framework for transparently migrating controller loads by applying real-time centralized cloud storage strategies to store data flow status and virtual connection management units [21].

In [22], the authors proposed a multi-dimension storage selection strategy to decide the optimum distributed storage location for flow tables in SDN networks. However, their work only considered the efficient backup and recovery of the flow tables, rather than the entire control plane, which is quite different in applications. Besides, there is no design of security and privacy considerations during the data backup process. Authors of [23] proposed a novel validation framework to verify the control plane's performance across various failure scenarios and multiple failure recovery strategies. They modeled the validation problem of the distributed control plane as a optimization problem, which was also a notable contribution.

Most of these works only consider the recovery issues of individual data errors and they seldom pay attention to the backup and recovery issues of the entire controller data. Meanwhile, the backup of controller data mostly relies on third-party organizations such as cloud storage, and lacks any

security considerations for backup data. Besides, in SD-WAN applications, data backup is a high-frequency operation, while recovery is a low-frequency operation. The existing backup methods did not consider reducing the communication cost of high-frequency backup operations to save the communication resource of control channels. We propose DDBR in this paper, an innovative data backup and recovery solution to address these limitations.

III. Distributed data backup method in SD-WAN

Traditional data backup and recovery solutions usually use dedicated storage devices or cloud services for remote disaster recovery. When this kind of methods are applied to SD-WAN controller backup, two problems may emerge. First, the data freshness cannot be guaranteed. Remote backup takes time and can only be carried out periodically, so it cannot meet the freshness requirements of the controller's dynamically changing data. Second, the remote backup data is usually still stored in centralized manner or third-party hosted, and the backup data faces availability and privacy risks. Therefore, this paper proposes a distributed data backup solution that maintains both data freshness and privacy.

As already mentioned, the data backup in SD-WAN needs to overcome the challenges caused by unreliable network environment, so the error-tolerant and correcting performance of the backup method are important. Recently, the distributed storage system erasure coding technology is widely used in the communication industry and it can solve certain error correction problems [24]. However, there are three problems when applying it to SD-WAN controller backup. 1) The encoding process causes the data length to increase, which introduces additional transmission overhead; 2) The decoding process often involves solving the inverse matrix, which has high computational complexity; 3) The original data has no security protection and is easy to be obtained by attackers.

In such a context, this work improves SSS and proposes DDBR, a new data backup framework. DDBR is not only tolerant to the unreliability of the WAN communication channels, but also ensures the security of backup data, and optimizes the design for application requirements of SD-WAN. This section introduces the DDBR framework.

A. Distributed Data Backup Architecture and Principle

The traditional SDN data plane is responsible for data forwarding, and the switches store only a small amount of information such as routing tables and flow statistics. With the rapid development of storage technology, the RAM of switches produced by mainstream manufacturers generally accounts for more than 4GB, and the storage capacity of hard disks usually is in excess of 32GB. These storage capacities far exceed the storage requirements of SDN data forwarding, enabling the switches to back up some control plane data. Meanwhile, the data transmission between the SDN controller and switches is implemented through dedicated encrypted tunnels, which are highly reliable. So it is feasible to use the switch to back up the controller data. The proposed framework is based on the switch storage for controller data backup. The overall architecture of the proposed data backup system is shown in Fig. 1.

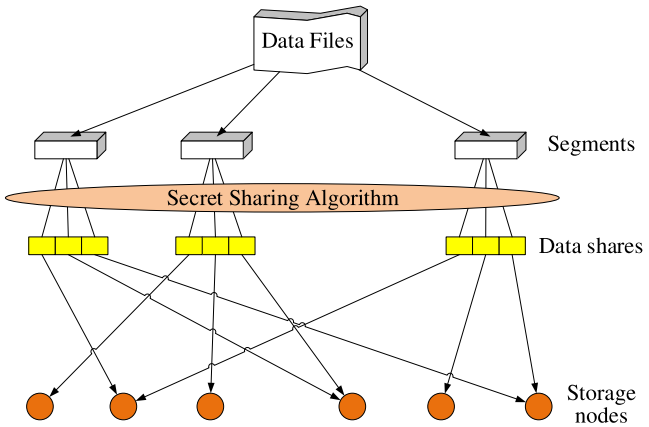

Fig. 1. System architecture of the proposed distributed data backup method.

In the control plane, the data files to be backed up are segmented into many data segments (e.g., r segments) with identical data length. After being encrypted and uniformly numbered, each data fragment are then divided into m data shares by implementing the secret sharing algorithm. Considering the actual needs for SD-WAN controller backup, we improve the original SSS algorithm in terms of two aspects. One is to encrypt the data segments first, and then implement the secret sharing algorithm, and the other is to add a unified number to the encrypted data fragments to facilitate their subsequent management. This is because there are many secrets to be shared rather than only one.

There are two considerations for introducing the encryption operation. One is that the original SSS requires that the sharers of the secret can self-organize to reconstruct the secret, while in a controller backup task, this is unsafe and unnecessary. The backup controller can perform decryption and restore the data uniformly. The encryption operation can ensure data integrity and avoid collusion within the compromised switches to obtain backup data. Since the mainstream asymmetric encryption algorithms have requirements for key length (usually longer than 256 bits), and generate long encrypted data blocks, which does not help in terms of data recovery and transmission efficiency. This architecture employs a one-time encryption method, and a key with equal length is added to each data block by XOR. This operation will not change the original data length, but it ensures confidentiality. The key can be pre-stored on the backup controller as offline backup data. Next, the data shares are ready to be stored in the data plane switches.

In the data plane, we assume that there are a total of k switches available. The controller randomly selects m switches and among them stores the data shares from the data segments, $m < k$. Note that for each data segment, the m storage node sets selected by the controller may be completely different. This is another adaptive improvement made to SSS. Typical SSS applications usually allocate a data share to each storage node by default. The advantage of this setting is that in all k switches, at most $k - m$ switches are allowed to fail to complete the backup operation due to link congestion or equipment failure, and the backup process will not be

affected. In addition, reducing the number of storage nodes can also reduce the channel resource overhead during the backup process. Under this setting, the probability of a single storage node storing the data shares of all r data segments will also be reduced to

$$p_u = \left(\frac{k}{m}\right)^r. \quad (1)$$

When r is large, this probability will be very small, which will further disperse the backup data and play a good role in data privacy protection.

B. DDBR Data Sharing Method

This subsection discusses in detail the DDBR specific methods of data sharing and reconstruction. Assuming that F is the file to be backed up. At the very beginning, F is divided into data segments of equal length l according to the actual needs, e.g., $l = 12$ bits for each fragment. Assume that the number of segments finally obtained is r and the segments are denoted by D_i, where $1 \leq i \leq r$. It does not matter if the length of the last segment is less than l. Then, F can be presented as:

$$F = D_1 D_2 \cdots D_r. \quad (2)$$

We execute the data sharing algorithm for each data segment separately. Without loss of generality, take D_1 as an example. Choose a key of length l as a one-time key, denote it by K_1, and perform XOR operation with D_1, then we get the secret s_1 to be shared.

$$s_1 = D_1 \bigoplus K_1. \quad (3)$$

Choose a prime number p, such that $p > s_1$ (treat s_1 as an integer). Both the secret sharing and reconstruction process will be completed in the p-element finite field. We take the (t, m) threshold scheme to share the secret. We need to split the secret into m shares, with at least t shares needed to reconstruct the secret s_1. To achieve this goal, we need to construct a polynomial of degree $t-1$ and take s_1 as the constant term [12]. For this we generate $t-1$ random numbers in the finite field, denote them as $c_1, c_2, \cdots, c_{t-1}$. Then we can build the polynomial as:

$$f(x) = s_1 + c_1 x + c_2 x^2 + \cdots + c_{t-1} x^{t-1} \mod (p) \quad (4)$$

We number all k switches as $1, 2, 3, \cdots, k$. Assuming that we randomly select m from the available switches as sharers of s_1, and denote their numbers by $v_1, v_2, \cdots, v_m$. Substituting them for x in (4), we get m data shares (in the sense of modulo p) as follows:

$$y_1 = f(v_1), y_2 = f(v_2), \cdots, y_m = f(v_m) \mod (p) \quad (5)$$

Now the controller can distribute $y_1, y_2, \cdots, y_m$ to $v_1, v_2, \cdots, v_m$ respectively as backup data.

By performing the process described above on all the data segments in sequence, the file F can be fully backed up. The most important difference between the proposed data sharing method and SSS is that we do not generate data shares for all switches, but randomly select m accessible ones. By doing this, on the one hand, redundancy is introduced to improve the fault tolerance of the backup system, in case of unreliable network condition. On the other hand, this approach reduces the communication cost to distribute m data shares instead of k ($m < k$). Since the backup nodes are randomly selected for each segment, it is unlikely that one node stores all the backup data segments. This reduces the risk of potentially hijacked nodes to steal the complete data set.

Algorithm 1: DDBR Data Sharing Algorithm

Input: One time key K; Key identifier α; Set of switches V; File F; Number of data shares m; Threshold number of recovery nodes t; Segment length l; Prime number p.
Output: Data shares for switches $Y_1, Y_2, \cdots, Y_k$.

1 Initialize $r = length(F)/l$, $Y_i = \phi, 1 \leq i \leq k$;
2 $D_1 D_2 \cdots D_r = Segment(F)$ by length l;
3 **for** $j = 1 : t-1$ **do**
4 $\quad C_j = Randnum()$;
5 **end**
6 **foreach** D_i **do**
7 $\quad s_i = D_i \oplus K(\alpha + (i-1) \cdot l + 1 : \alpha + i \cdot l)$;
8 $\quad$ Randomly select m switches $v_1, v_2, \cdots, v_m$ from V;
9 $\quad$ **foreach** $v_d, 1 \leq d \leq m$ **do**
10 $\quad\quad y_d = s_i + C_1 v_d + C_2 v_d^2 + \cdots + C_{t-1} v_d^{(t-1)} \mod (p)$;
11 $\quad\quad$ Attach y_d to Y_{v_d};
12 $\quad$ **end**
13 **end**
14 Distribute $Y_1, Y_2, \cdots, Y_k$ to $V_1, V_2, \cdots, V_k$ respectively;
15 **return**

The complete data backup process is shown in Alg. 1. Note that the one-time key used in the input part of the algorithm should be stored as static data on the backup server in advance. The algorithm receives the current key identifier α as input. After the file segmentation, the encryption process is performed for each data segment. The used key segment is intercepted according to the specified segmentation length l, and the bitwise XOR (Line 7) is implemented for encryption.

In the subsequent distribution process of data shares, the corresponding key identifier will also be distributed and backed up along with the data shares. The function $RandomNum(\cdot)$ refers to selecting random numbers as the polynomial coefficients of the secret sharing algorithm. Integers or real numbers can be randomly selected as needed, and the security of secret sharing algorithm is not affected. Generally speaking, considering that the modulo operation will be performed later, positive integers are usually selected. Note that these coefficients do not need to be backed up with data shares, and these coefficients will not be used in the data recovery process.

C. DDBR Data Recovery Mechanism

In the data recovery process, since the backup nodes are randomly selected for each data segment while sharing, there is no guarantee that each data segment can get at least t

shares, if only the backup data from t nodes is collected. Our solution is to collect the backup data of $k-m+t$ switches for data recovery. In this way, even if all the unselected $m-t$ switches have a share of the data segment, it can still be guaranteed that at least t data shares are collected. This method introduces higher requirement than the typical SSS for secret reconstruction. However, considering that data backup of the SD-WAN controller is a high-frequency operation, while data recovery is a low-frequency one, it is still cost-effective to have a slightly higher communication cost in the recovery process and a substantial cost reduction in the backup process.

After the backup controller finished collecting t data shares of D_1, the data recovery method can still use the classic Lagrange polynomials to find the constant $s_1 = f(0)$. Specifically, suppose that $v_{11}, v_{12}, \cdots, v_{1t}$ are the index values of the t collected switches, the following formula is used for calculation:

$$D_1 = \left(\sum_{i=1}^{t} y_i \prod_{1 \le j \le t, j \ne i} v_{1j}(v_{1j} - v_{1i})^{-1} \mod (p) \right) \bigoplus K_1 \tag{6}$$

It should be pointed out that the inverse operation in eq. (6) is the process of finding the inverse element on the p-element finite field. The complete process of data recovery is shown in Alg. 2. The main difference between this algorithm and the backup process is that the file and data segments are ordered in the backup phase. While after the backup controller extracts all the backup files from the switches, the data shares and files will become disordered. Therefore, the relevant index information must be backed up along with the corresponding data shares. The relevant details of the data index structure will be presented in the next section.

When performing recovery, the extracted data shares should be clustered according to the File ID, and all data shares belong to each cluster should be sorted into the same bucket (Line 2). In each bucket, all data shares need to be further clustered according to Segment ID to confirm which data segment it belongs to (Line 4). Then we can use the $Checksum$ field to verify the integrity of the data. Shares that fail the Checksum verification will be discarded. We select t shares from the cluster and perform data recovery calculations by (6) (Line 16). Then intercept the key according to the key identifier for decryption (Line 17). Finally, concatenate the recovered data segments into the original file in order (Line 19). This completes the entire data recovery process.

Algorithm 2: DDBR Data Recovery Algorithm

Input: Backups $Y_1, Y_2, \cdots, Y_k$; One time key K; Key identifier α; Set of switches V; Threshold number of recovery nodes t; Segment lengh l; Prime number p.
Output: File set F.

1 Extract data shares from $Y_1, Y_2, \cdots, Y_k$;
2 Cluster the extracted data shares by File ID into different buckets, $B_1, B_2, \cdots, B_n$;
3 **foreach** *Bucket* B_i **do**
4 Cluster data shares in B_i by Segment ID into different sets, $S_1, S_2, \cdots, S_r$;
5 **foreach** S_j **do**
6 **foreach** *Data shares in* S_j **do**
7 Calculate Checksum of the header;
8 **if** *Checksum incorrect* **then**
9 Discard the error data share;
10 **end**
11 **end**
12 Select t error-free data shares from S_j, say $y_1, y_2, \cdots, y_t$;
13 Find the IDs of the corresponding backup switches, say $v_1, v_2, \cdots, v_t$;
14 Extract Key Identifier form y_1, say α;
15 $K_j = K(\alpha + 1 : \alpha + l)$;
16 $s_j = \sum_{u=1}^{t} y_u \prod_{1 \le e \le t, e \ne u} v_e(v_e - v_u)^{-1} \mod (p)$;
17 $D_j = s_j \oplus K_j$;
18 **end**
19 $F_i = Concat(D_1, D_2, \cdots, D_r)$;
20 **end**
21 **return** $F = \{F_1, F_2, \cdots, F_n\}$

IV. The DDBR Dual Backup Framework

In actual SD-WAN applications, as a network operating system, the controller not only undertakes network equipment management, routing rule formulation, network statistics collection, but also supports upper-layer applications. Its software system is very complex and has a large amount of data. Therefore, it is impractical to rely on the above method to perform a full backup. In order to solve this problem, we must propose a practical backup framework.

A. Data Inventory

Before performing controller data backup, we must make an inventory of all data resources and classify them according to their dynamic characteristics and freshness requirements. The data consists of:

- Static and quasi-static data. This type mainly refers to the data that remains unchanged or does not change for a long period of time. It includes the controller operating system data, basic configuration files, network topology, network device management files, and other supported applications and related configurations. This type of data is the vast majority of all data to be backed up. In order to support the operation of the proposed backup architecture, quasi-static data should include the one-time encryption key, the switch index and prime number p.
- Dynamic data. This type of data mainly refers to data that continues to change in a short period of time, including device status, network statistics, routing rules, and the dynamic configuration parameters of related software and hardware, controller cache data, etc.

The purpose of data inventory is to clarify the backup requirements according to their dynamic characteristics, so as to design targeted backup strategies. For dynamic data, online

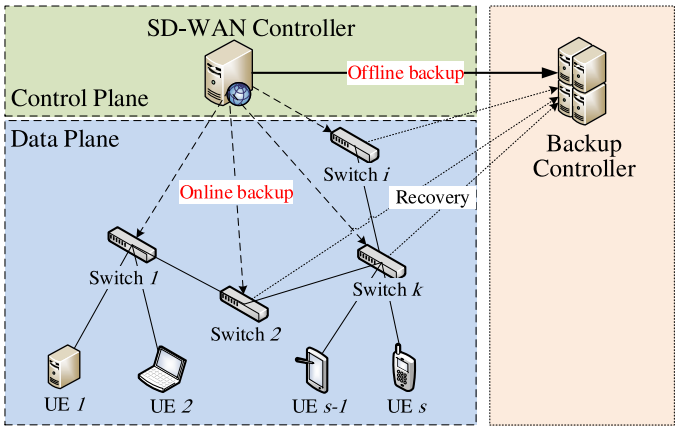

Fig. 2. The proposed data backup implementation framework.

backup methods should be adopted to improve its freshness, while for static and quasi-static data, regular offline backup methods are more suitable to save communication resources.

B. Data Backup Implementation Framework

According to the results of the data inventory, we design an online-offline dual backup implementation framework based on the SD-WAN architecture. The composition and working mode of the framework are shown in Fig. 2.

- Offline backup. The static data can be stored once at the beginning of the backup controller deployment. Since quasi-static data also has update requirements under a longer time scale, data can be synchronized regularly between the SD-WAN controller and the backup controller. In order to ensure security, offline backup methods are generally used for data synchronization.
- Online backup. For dynamic data, updates are required frequently. The online backup scheme is a combination of regular backup and condition-based backup. Condition-based backup is usually a partial data update, which is executed when the dynamic data file undergoes important changes. The original shares are replaced by the newly generated ones, and the data file stored by each switch remains unchanged. Regular backup is replacement of all data, which is performed when the network is relatively idle. Each switch will clear all the original data shares and receive new assigned backup data.

C. Data Structure

Since the original backup files are split into disordered data segments, a data indexing system must be constructed to facilitate the data recovery. The solution is to design a new data structure for backup packets. This structure not only considers the retrieval convenience and efficiency of data shares, but also considers the key identification requirements in the data recovery process and the integrity of the data fields. The structure of the backup data we designed is shown in Fig. 3.

This data structure is mainly composed of three parts: an index part, an integrity verification part, and a data share part. The index part should mainly consist of two fields, one is **File ID**, which is used to mark the file to which the share belongs. The proposed length is 12 bits, which can identify up to $4,096$ files. This field is long enough to cover the online backup task of current SD-WAN controllers. The other is **Segment ID**, which is used to identify the data segment corresponding to the share. The length is set to 20 bits, which can identify at most $1,048,576$ segments. Even if the length of each data segment is only selected as 12 bits, this field can still identify files up to 12 MB. For text messages, this file size is sufficient. If not, the length of each field can also be adjusted according to actual requirements. The relevant information can be backed up to the backup server as quasi-static data. The integrity verification part consists of two parts, which are 16 bits for **Key identifier** and 16 bits for **Checksum**. The **Key identifier** is used to mark the starting position of the Key (e.g., K_1 in (3)) which was used to encrypt the data share in the codebook. The maximum length that this field can identify is 65,536. The maximum length that this field can identify is 65,536. In practical applications, if the length is insufficient, it can be restarted from 1. This mechanism is similar to the fragment identification of IP packets, and the length of 16 bits is usually not likely to lead to confusion. The length of the **Data Share** part is the same to the size of data segments specified by the backup method.

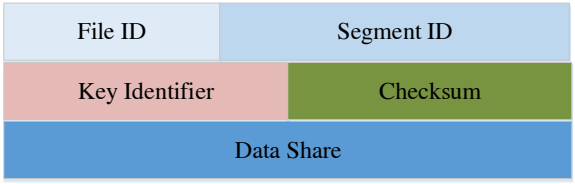

Fig. 3. The proposed data structure of backup packets.

When a single backup node stores a large amount of data shares, the proposed data structure of backup packets makes it possible to quickly identify and retrieve the file to which a specific data share belongs, accurately locate the key segment, and quickly recover the backed up data segment. The length of the extra packet introduced is only 64 bits, which is quite efficient. In addition, the introduction of checksum also adds an error discovery mechanism, which reduces the risk of backup data failure due to transmission errors. In practical applications, this data structure can be further streamlined for bandwidth resource constrained links by removing the key identifier and checksum filed, and only keeping the data index part. In this case, the starting position of the key needs to be recalculated based on the file ID and segment ID, and all backup files must be restored in order. The storage and communication resources are saved at the expense of some efficiency.

V. Performance Analysis

In this section, we will analyze the performance of DDBR in terms of backup success rate and recovery success rate.

A. Backup success rate

Recall the backup algorithm, for each data share s, we independently selected m from k routers to complete the backup. In the SD-WAN environment, due to the uncertainty of the network environment, the link between the control plane and the data plane may fail temporarily. This situation will cause the backup to fail. Since the backup process has no timeliness requirements, the controller can distribute data after confirming the network status of the target switch, or through the backup feedback information, to learn about the failure of the backup in time, so as to select another target switch to store the data share. In such case, the backup fails if and only if the number of routers available in the network is less than m.

Assuming that the average failure rate of the link between the control plane and the data plane is $1 \geq p_0 > 0$, and let the random variable X denote the number of unreachable routers, then the backup fails if and only if $X > k - m$. Since link failure events are independent, X obeys a binomial distribution with a mean p_0. Thus the backup failure rate can be formulated as

$$P(X > k - m) = \sum_{i=k-m+1}^{k} \binom{i}{k} p_0^i (1 - p_0)^{k-i}. \quad (7)$$

Therefore, the backup success rate is

$$P_b = 1 - \sum_{i=1}^{k-m} \binom{i}{k} p_0^i (1 - p_0)^{k-i}. \quad (8)$$

To save storage space, m is usually much smaller than k, and p_0 is also very small, so the backup failure rate is almost negligible. For instance, when $k = 20$, $m = 8$, $p_0 = 0.001$, $P(X \leq k - m) < 0.02$, the back up success rate $P_b > 0.98$. Therefore, our proposed DDBR has a vary high reliability.

B. Recovery success rate

For each data segment, the data recovery process needs to extract at least t error-free backup shares from m. We still assume that the random failure rate of the link is p_0, then data recovery will fail if and only if the number of reachable switches is less than t. That is, at most $m-t$ data shares among m cannot be collected to the backup controller. Similarly, due to the independence of link failures, the success rate of data recovery can be roughly estimated as

$$P(X < m - t) = \sum_{i=1}^{m-t-1} \binom{i}{m} p_0^i (1 - p_0)^{m-i}. \quad (9)$$

More accurately, assume that s storage nodes that are disabled in the recovery process. If $s \leq m - t$, the recovery is unaffected. The recovery success rate will be 1. Otherwise, $s > m - t$, the recovery fails if at least $m - t + 1$ data shares of one segment was distributed to these s disabled nodes. The probability of such events can be formulated as

$$P_d(s) = \sum_{j=m-t+1}^{s} \frac{\binom{j}{s}\binom{m-j}{k-s}}{\binom{m}{m}}. \quad (10)$$

Then the recovery failure rate of DDBR can be formulated as

$$P_f = \sum_{s=m-t+1}^{k} \left(\sum_{j=m-t+1}^{s} \frac{\binom{j}{s}\binom{m-j}{k-s}}{\binom{m}{m}} \right) \binom{s}{k} p_0^s (1 - p_0)^{k-s}. \quad (11)$$

Thereby the recovery success rate is

$$P_r = 1 - P_f. \quad (12)$$

Eq. (12) gives the accurate recovery success rate of each segment. Assuming that a file F has r segments, its recovery success rate may be more complicated since the success rates of segments is dependent. Generally, if r is large, P_r may not reach a satisfactory value. Therefore, the selection of the data segmentation length needs to consider its impact on the recovery performance to ensure that r is not too large, so that P_s remains within an acceptable range. In practical applications, since p_0 is usually very small, the impact of this accumulation of errors on the application effect is not significant.

VI. Performance Evaluation

In order to evaluate the effectiveness of the proposed DDBR backup solution, we apply it to a data backup instance, and evaluate its performance in terms of the average storage size on each node and backup success rate. We select two mainstream distributed backup methods for comparison: SSS and Blockchain, since they are similar to our method in terms of backup modes and security considerations.

A. The Storage Size

We select a file of 240 bits. The length of the data segment is set to $8, 10, 12, 16, 20$ bits, respectively. For our method and SSS, the data index structure adopts the simplified 32 bits version proposed in the previous section. The number of transactions in the Blockchain is set to 1 and the transaction input content is the file to be backed up. All backup data is stored in one block, and its data structure includes block size (4 bytes), header (80 bytes), transaction count (1 byte), transaction content (39 bytes, including version 4 bytes, input count 1 byte, input content 30 bytes, time 4 bytes). The experimental evaluation network employed consists a total of 10 backup nodes. We employed the $(6, 4)$ threshold scheme. After the selected file is stored in a distributed manner by the three methods, the average storage sizes of each storage node is illustrated in Fig. 4.

It can be noted that our proposed method DDBR occupies the smallest storage space of the backup node on average, and the storage size decreases with the increase in the segment length. In the Blockchain based method, since the file is not segmented, the storage size is not affected by the segment length. But the storage size is generally at a high level due to the fact that the block header is very long. For the SSS-based method, the average storage size decreases as the segment length increases, but the storage size is always significantly larger than in our method. It can be concluded that DDBR performs significantly better than the alternative methods in terms of storage occupation.

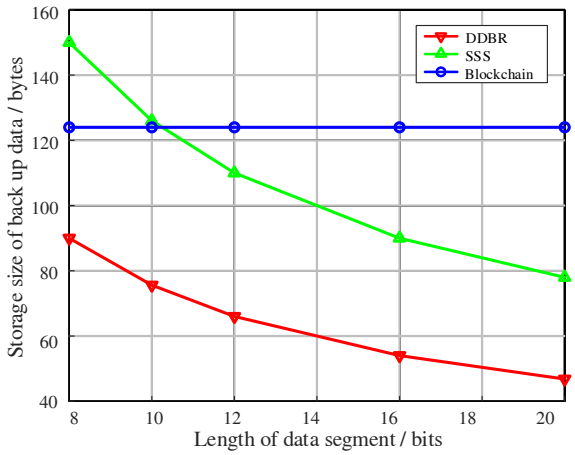

Fig. 4. Average storage size comparison of different segment lengths.

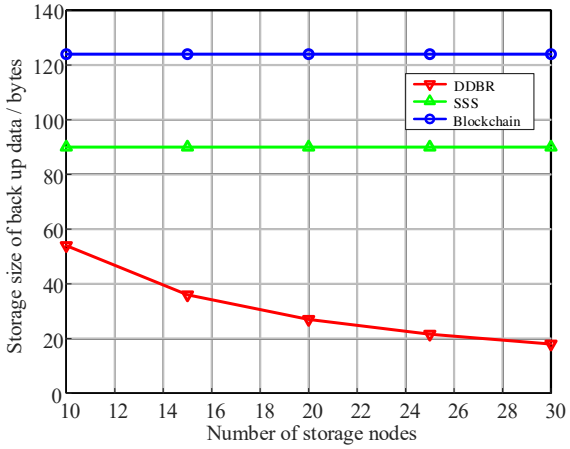

Fig. 5. Average storage size comparison of different number of storage nodes.

To verify the scalability of our proposed DDBR, we implement another experiment by setting the number of storage nodes range from 10 to 30. The segment length is fixed to 16 bits. Other parameter settings keeps unchanged. The results are shown in Fig. 5. As the number of storage nodes increases, the average storage size of SSS and Blockchain based methods are constant since both methods backup all the data shares to each nodes. While the storage size taken by our proposed DDBR decreases, which indicates the very good scalability performance of DDBR compared with benchmark methods.

B. The Communication Overhead

In order to verify the communication overhead of DDBR in the control data backup and recovery process, we designed another set of experiments. The length of the backup file we chose is still 240 bits as described in the previous section. The methods we use for comparison are still SSS and Blockchain based method. In the backup process, DDBR randomly selects 6 nodes to store backup data shares each time, while SSS and Blockchain based methods store backups to all nodes by default. In the recovery process, DDBR and SSS requires at least 4 backup copies of each data segment for recovery.

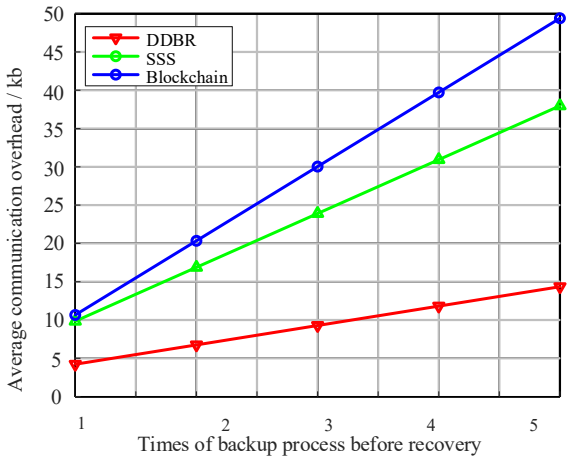

Fig. 6. Communication overhead of each node in multiple backups and a single recovery processes.

The length of data segment is fixed to 12 bits. Considering that in practical applications, the backup process is a high-frequency operation, while the recovery process is a low-frequency operation. Therefore, we have considered the case of performing a data recovery operation after 1, 2, 3, 4, and 5 backups respectively. The total communication overhead is used as the evaluation index. The experimental results are depicted in Fig. 6

As the number of backups before data recovery increases, the total communication overhead corresponding to all three methods are increasing approximately linearly. In the case of only one round of backup process, the Blockchain-based method is better than SSS in total communication cost. But when implementing more than 2 rounds of backups before recovery, SSS showed more and more significant advantages. The main reason for this results is that although the backup overhead of the Blockchain is high, the recovery overhead is relatively low. Data recovery can be completed by extracting only the backup data of one node in Blockchain based method, while SSS requires four. In any case, DDBR is significantly better than SSS and Blockchain based methods in the total communication overhead of backup and recovery processes. Especially with the increase in the number of backups, this advantage is still expanding. These results show that our proposed DDBR has significant advantages over the baseline method in terms of total communication overhead during data backup and recovery processes.

Next, we consider the total communication overhead introduced by the backup process to the whole network. We set the number of storage nodes range from 10 to 30 and the segment length to 16 bits again. We implement 2 backup processes and 1 recovery process. The communication overhead evaluation results are illustrated in Fig. 7. As the scale of the network increases, the total overhead of SSS and Blockchain based methods increase linearly. While the overhead of our proposed DDBR keeps constant. That is because we constantly select m storage nodes for each data share, which has nothing to do with the scale of the whole network. This verifies again that DDBR has very good scalability performance in real applications.

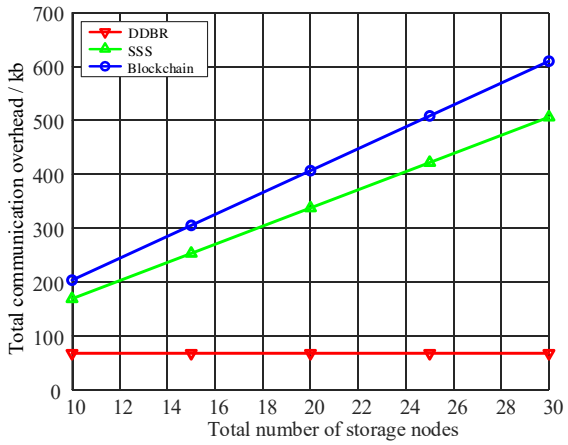

Fig. 7. Total communication overhead of the whole network in 2 backups and a single recovery processes.

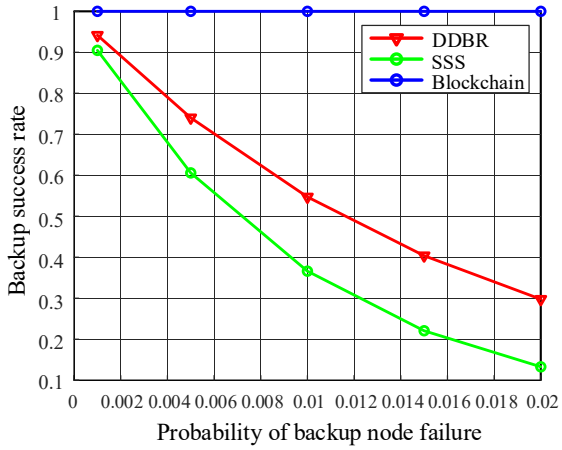

Fig. 8. Backup success rate comparison in different probabilities of node failure.

C. Backup and Recovery Success Rate

Consider that due to network congestion or equipment failure in the wide area network, data shares may not be transmitted to their backup nodes as planned during the backup process, so the backup success rate must be considered. The backup is successful if and only if all switch nodes receive their backup shares, as planned. This is very strict for backup algorithms with data segmentation. Blockchain-based method stores a complete file copy to each node, which ensures its high backup success rate with huge overhead. So only when more than half of the backup nodes are disconnected, the backup process based on Blockchain will fail. We continue to assume that there are totally 10 backup nodes in the whole network, and DDBR needs to distribute data shares to 6 randomly selected nodes in the backup process each time, while SSS distributes the data shares to all the backup nodes.

We choose the segment length to be 16 bits, so there are 15 segments in total. Assuming that the event of node failure is random, consider five cases of failure probability: 0.001, 0.005, 0.01, 0.015, and 0.02 respectively. The corresponding backup success rates are shown in Fig. 8.

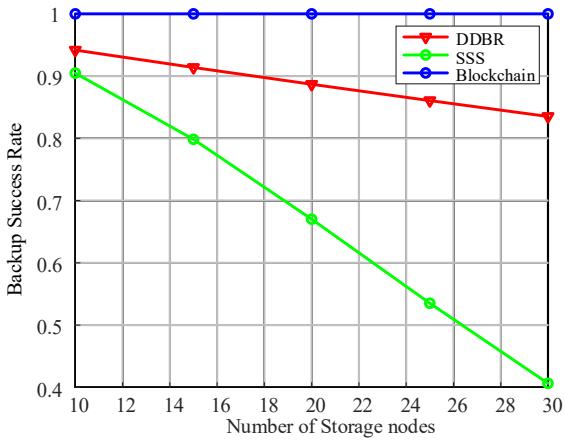

Fig. 9. Backup success rate comparison in different numbers of storage nodes.

Since we assume that the node failure rate is very low (the highest is not more than 2%), blockchain-based (marked by the blue broken lines) backup method can back up data with a success rate close to 100%, so its performance on this index is significantly better than DDBR and SSS. But please note that this superiority is based on huge storage and communication costs. For other two methods, as the probability of node failure increases, the backup success rate is significantly reduced. Due to the small number of backup nodes selected each time, our method is consistently superior to SSS in terms of backup success rate under various node failure probabilities.

We also studied the scalability performance. The results are illustrate in Fig. 9. Benefiting from the huge communication and storage cost, Blockchain based method still performs best, and DDBR is the second. However, as the scale of network increases, the backup success rate deteriorates rapidly. The decline in the backup success rate of DDBR is relatively flat, which shows that DDBR has stronger adaptability to the growth of network scale.

To sum up, the evaluation results in actual backup instance show that DDBR has very good performance, not only in terms of single-node storage size, but also the communication overhead in backup and recovery process, as well as the backup success rate and scalability.

VII. Conclusions and Future Work

In order to reliably backup the controller data in SD-WAN and improve network availability, we have proposed DDBR, an innovative distributed data backup and recovery method based on a secret sharing algorithm. This method first divides the data file into segments, and then divides each segment into several data shares and stores them in the switches. To reduce the amount of data storage and communication overhead, we propose an online-offline dual backup framework. Offline backup adapts to the static and quasi-static data, while the dynamic data are backed up to some randomly selected storage nodes online. We have also introduced an encryption mechanism to further enhance data confidentiality, and designed an index structure to facilitate the rapid data recovery. Future work will mainly focus on the detailed implementation of the

backup scheme in a prototype system and further optimization of storage efficiency.

ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China (2018YFE0205502).

REFERENCES

[1] Y. Zhang, C. Xu and G. Muntean, "A Novel Distributed Data Backup and Recovery Method for Software Defined-WAN Controllers," *IEEE Global Commun. Conf. (Globecom)*, Madrid, Spain, Dec. 2021.
[2] S. Ghosh, M. Iqbal, T. Dagiuklas, "A centralized hybrid routing model for multicontroller SD-WANs". *Trans. Emerg. Telecommun. Technol.*, vol.32, no. 6, e4252, May 2021.
[3] R. Scott Raynovich, "SD-WAN Growth Report", Futuriom, Jun. 2020.
[4] F. Benayas, Á. Carrera, M. García-Amado and C. A. Iglesias, "A semantic data lake framework for autonomous fault management in SDN environments." *Trans. Emerg. Telecommun. Technol.*, vol. 30, ett.3629, May 2019.
[5] Z. Yang and K. L. Yeung, "SDN Candidate Selection in Hybrid IP/SDN Networks for Single Link Failure Protection," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 312-321, Feb. 2020.
[6] K. Qiu, J. Zhao, X. Wang, X. Fu and S. Secci, "Efficient Recovery Path Computation for Fast Reroute in Large-Scale Software-Defined Networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1755-1768, Aug. 2019.
[7] J. Zheng, H. Xu, X. Zhu, G. Chen and Y. Geng, "Sentinel: Failure Recovery in Centralized Traffic Engineering," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 1859-1872, Oct. 2019.
[8] A. K. Singh, S. Maurya, Na. Kumar and S. Srivastava, "Heuristic approaches for the reliable SDN controller placement problem." *Trans. Emerg. Telecommun. Technol.*, vol. 31, e3761, Oct. 2019.
[9] T. Sasaki, A. Perrig and D. E. Asoni, "Control-plane isolation and recovery for a secure SDN architecture," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, Seoul, South Korea, Jun. 2016 , pp. 459-464.
[10] L. X. Yang, K. Huang, X. Yang, Y. Zhang, Y. Xiang and Y. Y. Tang, "Defense against advanced persistent threat through data backup and recovery," *IEEE Trans. Network Sci. Eng.*, early access article.
[11] X. Ren, G. S. Aujla, A. Jindal, R. S. Batth and P. Zhang, "Adaptive Recovery Mechanism for SDN Controllers in Edge-Cloud supported FinTech Applications," *IEEE Internet Things J.*, early access article.
[12] Adi Shamir., "How to share a secret". *Commun. ACM*, vol. 22, No. 11, pp. 612–613. Nov. 1979.
[13] R. Bitar and S. E. Rouayheb, "Staircase Codes for Secret Sharing With Optimal Communication and Read Overheads," *IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 933-943, Feb. 2018.
[14] W. Huang, M. Langberg, J. Kliewer and J. Bruck, "Communication Efficient Secret Sharing," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 7195-7206, Dec. 2016.
[15] M. Soleymani and H. Mahdavifar, "Distributed Multi-User Secret Sharing," *IEEE Trans. Inf. Theory*, vol. 67, no. 1, pp. 164-178, Jan. 2021.
[16] P. C. Fonseca and E. S. Mota, "A Survey on Fault Management in Software-Defined Networks," in *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2284-2321, 2017.
[17] Y. Yu et al., "Fault Management in Software-Defined Networking: A Survey," in *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 349-392, 2019.
[18] M. Reitblatt, M. Canini, A. Guha, N. Foster., "Fattire: Declarative fault tolerance for software-defined networks," in *Proc. 2nd ACM SIGCOMM HotSDN*, pp. 109-114, Aug. 2013.
[19] N. Katta, H. Zhang, M. Freedman, J. Rexford., "Ravana: Controller fault-tolerance in software-defined networking," in *1st ACM SIGCOMM Symp. SDN Res.*, Santa Clara, CA, USA, Jun. 2015, pp. 1-12.
[20] M. Mohiuddin, M. Primorac, E. Stai and J. Le Boudec, "FCR: Fast and Consistent Controller-Replication in Software Defined Networking," *IEEE Access*, vol. 7, pp. 170589-170603, 2019.
[21] K. Basu, A. Hamdullah and F. Ball, "Architecture of a Cloud-based Fault-Tolerant Control Platform for improving the QoS of Social Multimedia Applications on SD-WAN", in *Proc. 13th Int.Conf. Comm. (COMM)*, Bucharest, Romania, Jun. 2020, pp. 495-500.
[22] W. Ren, Y. Sun, T. Wu and M. S. Obaidat, "A Hash-Based Distributed Storage Strategy of FlowTables in SDN-IoT Networks," *IEEE Global Commun. Conf. (Globecom)*, Singapore, Dec. 2017, pp. 1-7.
[23] J. Xie, D. Guo, C. Qian, L. Liu, B. Ren and H. Chen, "Validation of Distributed SDN Control Plane Under Uncertain Failures," in *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1234-1247, Jun. 2019.
[24] M. Sipos, J. Gahm, N. Venkat and D. Oran, "Network-Aware Feasible Repairs for Erasure-Coded Storage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1404-1417, Jun. 2018.

Yi Zhang received the M.S. degrees in Electronics from Mid Sweden University, Sweden. She is currently pursuing her Ph.D. degree in Institute of Network Technology, Beijing University of Posts and Telecommunications. Her research interests include network security, architecture design and robustness analysis of network topology, network controllability, etc.

Lujie Zhong received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2013. She is currently an Associate Professor with the Information Engineering College, Capital Normal University, Beijing. Her research interests include communication networks, computer system and architecture, and mobile Internet technology.

Shujie Yang Shujie Yang (Member, IEEE) received the Ph.D. degree from the Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China, in 2017, where he is currently a Lecturer with the State Key Laboratory of Networking and Switching Technology. His major research interests are in the areas of wireless communications and wireless networking.

Gabriel-Miro Muntean (Senior Member, IEEE) is a Professor with the School of Electronic Engineering, Dublin City Univ. (DCU), Ireland, and co-Director of the DCU Performance Engineering Lab. He has published over 450 papers in top international journals and conferences, authored 4 books and 22 book chapters, and edited 7 other books. His research interests include quality, performance, and energy issues related to rich media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. He is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communications Surveys and Tutorials, and a reviewer for top international journals, conferences, and funding agencies.