Blockchain-Based Controller Recovery in SDN

Sudip Misra[1], Kounteya Sarkar[2], and Nurzaman Ahmed[3]

[1,2,3]Department of Computer Science & Engineering
Indian Institute of Technology
Kharagpur, India-721302
[1]sudipm@iitkgp.ac.in, [2]kounteya1000@gmail.com, [3]nurzaman@cse.iitkpg.ac.in

***Abstract*—In this paper, we propose a Blockchain-based solution for the recovery of an SDN controller back to a previously known state upon sudden failure. A lightweight minimal Blockchain ledger containing metadata details about each controller event is maintained by the switches. The set of all instructions given by the controller to the switches denotes the state of the controller at that instant. Whenever a new event occurs, the meta-information about it gets stored in the Blockchain which is updated in the switches after regular epochs. Upon sudden failure and subsequently coming back online again, the controller downloads all the tables and information from the respective switches. It checks and compares the metadata contained in the Blockchain with those data received from the switches. In addition to the existing security services provided by Blockchain, the proposed scheme can further solve the controller failure problem. The performance of the proposed solution is measured through simulation. The proposed scheme with the metadata-based solution saves about 75% of space and a controller can securely recover with a duration of 50 Sec.**

***Index Terms*—Software Defined Network (SDN), Blockchain, Controller Failure, Internet of Things (IoT)**

I. Introduction

Software-Defined Networking (SDN) allows us to separate the data plane from the control plane offering new and exciting features in modern networking [1]. With only the data plane residing in the switches (in SDN we use 'switch' to mean both traditional switches and routers) and the control plane moved up to a centralized controller. Thus, SDN allows us to build dynamic network typologies on-demand with required characteristics. There exists specific communication protocols such as [2] and [3] to program the switches regarding forwarding decisions. We consider the controller to be central to the entire SDN architecture. As in any similar architecture with a central component, SDN suffers from central point-of-failure. Recovery of the controller from a failure is thus of paramount importance for the continuous operation of the network. BlockChain was first introduced by Haber and Stornetta [4] in 1991 and popularized by Nakamoto *et al.* [5] in 2008 with the Bitcoin cryptocurrency application. It has evolved manifolds since then to include a lot from cryptocurrencies and crypto-tokens (such as Bitcoin and Ethereum [6]) to secure distributed applications (DApps) and smart contracts [7] and [8].

In this work, we propose an approach towards the failure recovery of an SDN controller using Blockchain-based distributed ledgers. Blockchains are inherently immutable data structures, and thus, provide secure and verifiable platform for various applications. Although the two concepts of SDN and Blockchain are inherently different, one being a fully centralized approach while the other being distributed in nature, we show the orchestration of both for failure recovery while maintaining the basic characteristics of both SDN and Blockchain. Our approach is aimed at effectively utilizing the security provided by the distributed Blockchain ledger containing various information which is immutable and verifiable. These characteristics ensure authentic recovery of the controller to a previously known state upon failure, assuming that the controller has no previous knowledge of its states before failure. In summary, the following are the *contributions* of this paper:

- A lightweight Blockchain-based SDN architecture concerning the resource limitations of SDN switches is proposed. Instead of using the complete Blockchain ledger, the switches use a minimal ledger across the network.
- A Blockchain-based solution for recovery of an SDN controller back to a previously known state upon sudden failure is proposed. The metadata details about each controller event are stored in the switches, which further help the controller in the recovery processes.

The rest of the paper is organized as follows. Section II gives the literature survey and related works on SDN recovery and similar concepts. Section III gives a detailed explanation of the proposed idea. Performance evaluation and analysis of the proposed scheme is presented in Section IV. Finally, Section V concludes the paper.

II. Related Works

Link failure and controller recovery in SDN is an active area of research with many publications existing in the literarure. Adrichem *et. al* [9] proposed a fast failure recovery mechanism using bidirectional link states and OpenFlow communication. Chu *et. al* [10] has addressed the problem of single link failure in hybrid SDN architecture using tunnelling in IP. Through simulation they have concluded that their design requires only small number of SDN switches. Capone *et. al* [11] uses OpenState, an extension of OpenFlow to promote instantaneous and immediate recovery in SDN with no packet loss. Many works can be further found in the literature on Blockchain applications over SDN networks. Sharma *et al.* [12] proposed DistBlockNet, which uses Blockchain to impart

security to SDN networks for the Internet of Things (IoT) scenario. Basnet and Shakya proposed a BSS protocol to provide security to SDN networks and architectures using Blockchains [13]. Houda *et al.* [14] proposed an algorithm ChainSecure. Alharbi *et al.* [15] gives a comprehensive survey paper on the various usage and deployment of Blockchain in SDN. SDN architectures, as the traditional ones, also suffer from faults, especially single point of failures because of their inherent centralized architecture. Liang *et al.* [16] proposes a recovery mechanism using blockchains for distributed controllers in software defined optical networks. This proposal however considers an architecture of several controllers distributed in a hierarchy with controller based ledgers. Khan *et al.* [17] proposes a blockchain based recovery scheme for failed nodes in a Wireless Sensor Network scenario with low power nodes having wireless communication. Kuzniar *et al.* [18] proposed AFRO, an OpenFlow based failure recovery model for SDNs. Mohan *et al.* [19] proposed two proactive algorithms for efficient SDN failure recovery based on rerouting. Fang *et al.* [20] addresses the central point of failure problem by proposing a Fast and Load-aware Controller FlohSequenceNumberInstructionTypcalailover (FLCF) mechanism. To the best of our knowledge, a Blockchain-based bottom-up controller recovery (i.e., switches to controller) to recover a failed controller has not been proposed before.

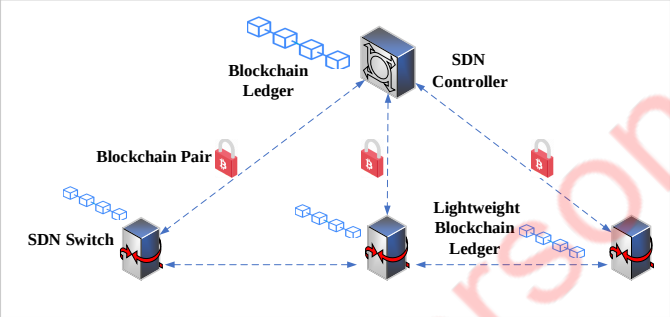

Fig. 1. The proposed Blockchain based SDN

The existing Blockchain-based solutions in SDN do not consider the space limitation in the switches. Secure controller recovery in SDN is still challenging.

III. Blockchain Based Controller Recovery

This section of the paper discusses the proposed Blockchain based controller recovery scheme.

A. SDN Controller Recovery Scheme

In a typical SDN scenario, as shown in Fig. 1, multiple switches are controlled by a single SDN controller. Hence, the entire architecture faces threat from single-point failure. This means that upon the failure of the controller the network collapses , as the switches are no longer able to consult the controller. In this paper, we try to resolve this issue by proposing a bottom-up Blockchain-based approach. If the controller suddenly fails, the switches cooperate to recover the controller to a previously known state. This is why this

Timestamp
Instruction_Serial_Number
Destination_Switch_ID
Instruction_Type
Destination_Switch_Sequence_Number
Instruction_Type_Sequence_Number

Fig. 2. Different fields of an instruction metadata of messages

is a bottom-up approach. A distributed Blockchain exists across the switches which helps in authenticating the recovery process so that the controller can safely recover itself to a correct previous state. In the following paragraph we explain the recovery process that follows once the controller fails including importantt notations:

Let S_t denote the state of the controller at time instant 't'. The state of the controller denotes the collection of all the various flow rules that the SDN controller maintains across the switches that it controls. Let S_f denote the state of the controller at the time instant immediately before the controller fails. The objective of the recovery algorithm is to make the controller recover back to the state S_f upon failure. We define a controller event as any event 'e' which makes some change to the state S_t. This means that, whenever the controller gives some new instruction to the switches or modifies existing instruction a controller event is triggered. Every such controller event is associated with some metadata along with the actual data of the event. We will not consider the actual data of the event, for example, any new instruction to the switches, rather we will consider only the metadata which will be stored in the Blockchain. This is because during the recovery process the controller will get the actual event data from the respective switches themselves. It will only consult the metadata stored in the Blockchain to authenticate the recovery process.

Since this is an initial presentation of the idea of Blockchain-based recovery without loss of generality. Fig. 2 shows the metadata associated with each controller's events. The fields are all self-explanatory and have their usual meanings. *Destination_Sequence_Number* signifies the sequence number of the particular instruction with respect to the destination switch while *Instruction_Type_Sequence_Number* denotes the sequence number concerning the particular type of instruction, such as add new flow entry or modify flow table among others.

We now state the sequence of steps during the recovery process. Upon sudden failure of the controller, the switches are unable to get any further instructions. Thus, these switches hold the latest instructions from the controller which they have last received. We assume that the controller upon failure loses all its state information S_t that it holds. Upon coming back online, (without loss of generality, we assume that the controller comes back online within a reasonably finite amount

of time) the controller has no information about state S_f, when it failed. Hence, it initiates the following sequence of steps to recover back to the state S_f.

1) The controller sends a *provide_last_info* request message to all the switches that it is connected to, giving the timestamp when it came back online.
2) The switches upon receiving the request from the controller, each of them sends their flow table entries that they received last from the controller.
3) The controller performs a union operation on all the flow tables that it receives from all the switches. This union represents the set of flow rules that the controller had immediately before failure, i.e the state S_f.
4) The controller must, however, authenticate the above-mentioned process of recovery. This is because since the controller has no previous knowledge of the flow tables and switch configurations it must be sure of the authenticity of the flow tables that it receives from the switches.
5) The controller thus consults the local Blockchain of any switch under it. The choice of the switch is random to remove any biasness. The Blockchain contains the metadata of every previous controller event (refer Fig. 2) which the controller can easily compare with the actual flow table switches and verify their authenticity.
6) The immutable nature of the Blockchain ensures that the controller is successfully able to verify and authenticate the flow tables that it receives from the switches. Any invalid or malicious flow table rule will not have its metadata on the Blockchain enabling the controller to choose only valid rules.
7) In this process any malicious switch will fail in its purpose of sending corrupt data to the controller, as then it will have to first compromise the Blockchain to place its own faulty metadata on the chain. As it is well established the security of the Blockchain prevents this from happening. Only valid switches will be able to participate.
8) Thus, the controller will have the set of all flow rules from all the switches, and after verification through Blockchain, it will recover itself back to the state S_f.

Following the list of sequences stated above the controller can recover itself back to the state S_f. Here we state a very important assumption that we have made in our solution. In an ideal SDN scenario, the switches may not be aware of each other or how many switches are there in the network. However, in our proposal, we assume that all the switches under the said controller know at least the identities of all the other switches, as the switches run a Proof-of-Authority (PoA) consensus algorithm among themselves to add new blocks. PoA consensus algorithm requires that the participating entities know each other to verify the blocks. The switches, however, need not be aware of the path to another switch from itself as they can simply send out broadcast flood messages to communicate about Blockchain related issues. Fig. 3 shows

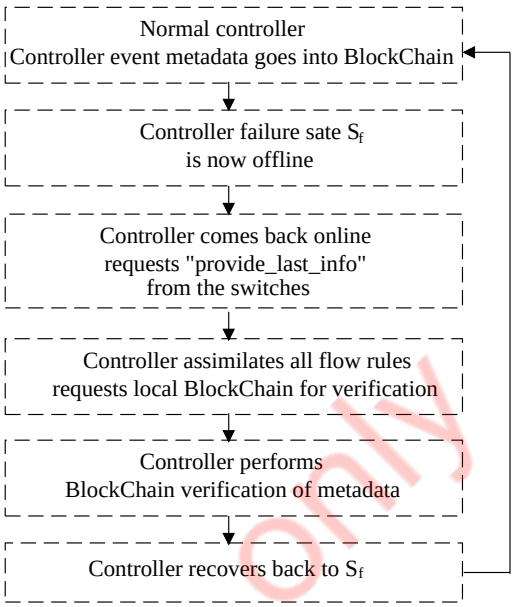

Fig. 3. Sequence of states of the Controller

the sequence of the recovery process listing the various states of the controller. Algorithm 1 shows the algorithm from the perspective of the SDN switches of the process they follow collectively in case of controller failure

Algorithm 1: Controller recovery mechanism

Inputs: Controller C at state S_t, set of SDN switches S

Outputs: Controller C at state S'_t, $t' > t$, set of SDN switches S

for *Each switch* $\in S$ **do**
  Pool the controller C using regular pings
  /* Ping wait duration is left open for implementation according to requirement */
  if *Ping_response* == *Normal* **then**
    /* Ping_response refers to the response of ping by controller C */
    S_t = Normal
    Record controller event metadata on blockchain
  else
    S'_t = Fail
    Initiate Blockchain-based recovery
    do until $S_t = S_f$
  Controller C recovers back to state S_f
  $S'_t = S_f$

B. The Lightweight Blockchain Solution

In this section, we detail the nature of the Blockchain, the structure of each block and the PoA based consensus algorithm

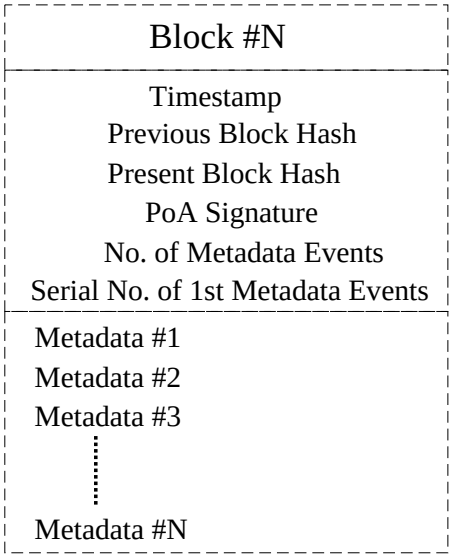

Fig. 4. Structure of each block of the Blockchain

that the switches run. Fig. 4 gives the structure of each block in the Blockchain. Each block consists of block parameters like timestamp, a hash of the previous block, PoA signature by the respective authoritative node and likewise along with a series of metadata about controller events. The structure of metadata is already shown in Fig. 2. We state below in points the procedure of block creation and Blockchain formation.

- Upon the occurrence of every controller event, which has been defined before, the controller sends the metadata of the event to every switch under itself.
- Using public-key cryptography, each of the metadata entries in a single block is signed by the controller, which can be verified by other entities. The intended recipient switch for every controller message similarly signs the metadata against the controller's message and broadcasts the signed metadata to every other switch. It may be noted that the switch only sends the signed metadata and not the actual data.
- The other switches, upon receipt of the signed metadata from the recipient switch, verify the metadata against the same metadata signed by the controller. In this way, every switch can verify the controller events.
- Once a fixed number of controller events has been verified, the authoritative switch at that instant collects these metadata into a new block, gives the timestamp and hash value, previous block's hash value and its PoA signature to create the block and broadcasts it. Upon receipt of the block, the other switches add it to their Blockchain as the latest new block since it has already been signed by the authoritative block.
- We propose for the sake of simplicity that the authoritative switch for every new round gets selected by a round-robin fashion. The Blockchain network is private because it exists only across the SDN switches and is restricted

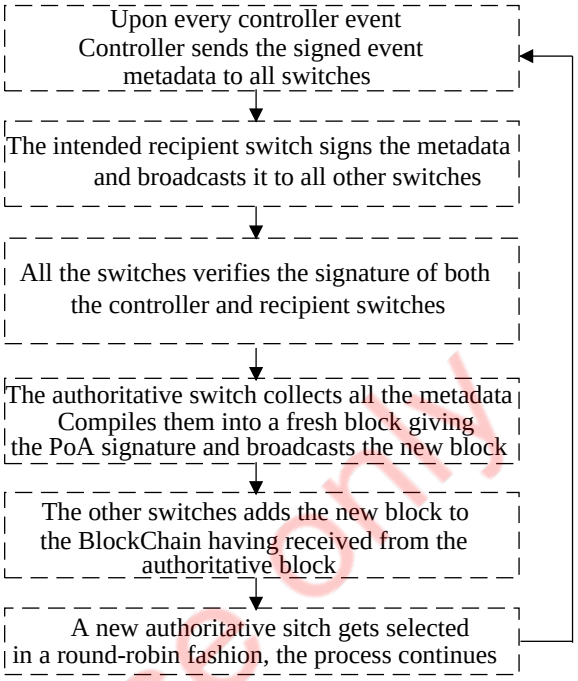

Fig. 5. Sequence of events in Blockchain formation

because only the controller and the switches themselves are the entities that can access them. Being a preliminary work in this aspect, it is reasonable enough to choose the next authoritative switch in a round-robin fashion.

The event sequence formation process and details of the corresponding events are shown in Fig. 5. We choose to use a PoA based consensus, because in the normal scenario there are only a limited number of switches under a controller. Therefore, going for a PoW based consensus-like Bitcoin wastes a lot of the switches' processing power to compete for consensus with only a few members, or that we would need special Application-Specific Integrated Circuits (ASICs) to be integrated with the switches for competitive consensus. Furthermore, the Blockchain is a private and special-purpose only to help in controller recovery upon failure with no commercial or profit-motive like Bitcoin. Thus there is no requirement to reward any particular switch by giving crypto tokens and currencies upon a successful PoW competition and mining (in Bitcoin terminology). Instead, in our proposed round-robin PoA based consensus, each of the switches in the network has equal and shared responsibility of successfully creating and adding the next block to the Blockchain.

IV. Performance Evaluation

In this Section, we show the performance evaluation of the proposed architecture. We performed simulation with the help of Mininet using one controller and switches under it. Our simulation shows that our proposed Blockchain solution can recover back the controller using the process described above. We first give a theoretical study as to why our model is feasible

and will always allow the controller to recover. It may be noted that at any time instant, S_t denotes the state of the controller which is the collection of all the flow rules across all switches. The state S_f can be calculated as:

$$S_f = \bigcup_{i=1}^{N} F_i^f \tag{1}$$

where F_i^f denotes the set of flow table rules in a particular switch i at the point when the controller fails. The metadata of all the flow rules, of course, is put into the Blockchain. Upon failure of the controller and its subsequent recovery online, the controller requests flow table entries from all the switches. Since all the switches under the controller send their respective flow table entries to the controller, and assuming that state F_i^f is as described before, the following equation describes this process,

$$F_1^f \cup F_2^f \cup F_3^f \cup F_4^f \cup ..... \cup F_N^f = S_f \tag{2}$$

We see that the result of the union operation is the state S_f which is exactly what the controller wants to recover back. Finally, to complete the process, the controller asks any one of the switches for its local Blockchain, which it refers to verify the metadata contents according to the following logic equation,

$$\forall i, (Metadata_i) \wedge (Blockchain_i) \implies (Valid_i) \tag{3}$$

The index 'i' runs for all the flow rules across all the switches and the above equation signifies that for a particular flow rule, if the controller matches the metadata of that rule in the set S_f as well as correspondingly in the Blockchain, then 'i' is a valid flow rule and the controller accepts it. Hence, we see that the controller chooses only those flow rules, whose authenticity it can verify from the Blockchain.

Fig. 6 shows a sample Mininet SDN architecture with one controller and four SDN enabled switches. We simulated the proposed mechanism on such types of architectures and successfully got the desired output of the controller from its recovery back. To emulate the controller failure behavior, we programmed the controller to go unresponsive for a random duration at a random point of time. All the flow rules held by the controller got removed to mimic a failed controller. Upon becoming responsive again, the controller was made to follow the recovery algorithm to recover itself back. Fig. 7 shows the active state of the controller during the simulation time. On Y axis, value 1 refers to active and value 0 refers to controller failure. As can be seen, the controller recovers back to active state after certain time.

Finally, we give the justification of using metadata over the actual flow rule data. In the SDN scenario, the switches are expected to be independent without knowing each other's flow rules. Using the complete flow rules in the Blockchain violates this as then the switches are aware of the flow rules of every other switch. Hence, we store only the metadata. The

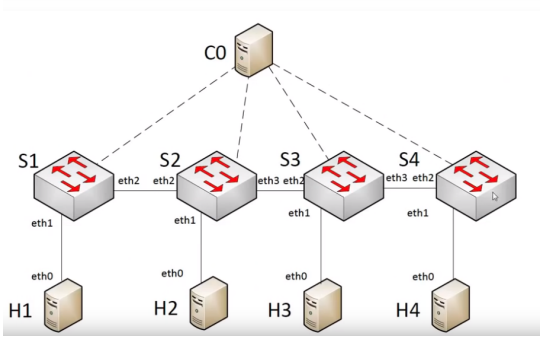

Fig. 6. Mininet architecture of SDN controller and switches

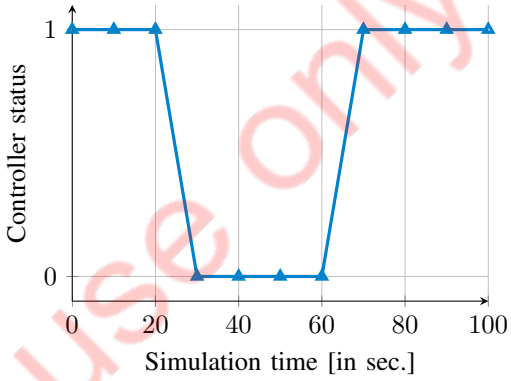

Fig. 7. Controller active status vs simulation time

metadata does not reveal the flow rules, yet at the same time can be used for verification of flow rules. The second most important aspect is the size of blocks. In our scheme, the size of the metadata (Fig. 2 shows the metadata structure) is at most 30 bytes considering the standard size of data types offered by most modern programming languages. On the other hand, the full length of average flow rule messages of OpenFlow protocol is about 200 bytes (it can be even very large depending on the type of message). Clearly, the use of only metadata saves about (100 - $\frac{30}{200} \times 100$) = 75% of total space. Considering a block of fixed size 'N', we can put in more metadata entries per block than actual flow rule entries. This enables the switches to save a lot of valuable space. Otherwise, they would have unnecessarily spent enormous spaces on just storing the Blockchain, clearly a wastage of the switches' computing and storage resource. Fig. 8 shows clearly the comparison between the space consumption by original flow rules (e.g. OpenFlow flow rules) vs our proposed use of metadata. The vast difference in space consumption is evident and visible from the comparison graph. Hence, considering the above two facts we have proposed using only flow rule metadata in the Blockchain.

V. Conclusion

This paper presented a Blockchain-based proposal to recover a failed controller using a bottom-up approach from the switches. The distributed Blockchain resides with the switches

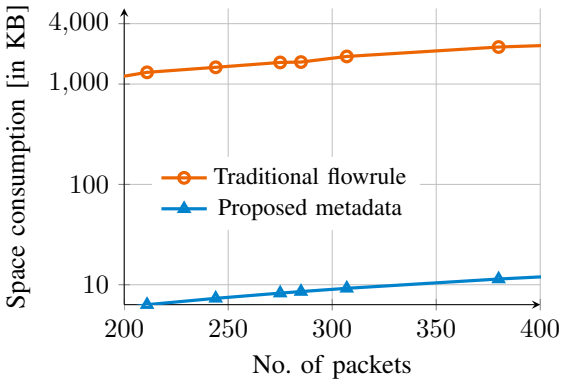

Fig. 8. Space consumption with increasing number of packets

and contains all controller event metadata. The switches run a round-robin PoA based consensus algorithm to decide on the next block. Upon failure, the controller receives the flow rules from individual switches and verifies them with the Blockchain metadata. We provided some mathematical equations to validate our proposal. Finally, we simulated our recovery mechanism. This interesting approach given by our proposal has a lot of areas to be explored in the future. In this work, only controller failure is considered assuming that the switches and the links remain intact. The next step is to make the scheme more realistic by incorporating other failures such as link and switch failures and how the Blockchain handles those situations. We believe that our proposed scheme will be accepted as a potential solution and open new areas of development.

Acknowledgment

The authors gratefully acknowledge the funding support received from SERB/IMPRINT-II (Sanction letter no SERB/F/12680/2018-2019;IMP/2018/000451, Dt. 25-03-2019) for executing parts of this work.

References

[1] W. Stallings, *Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional, 2015.
[2] "OpenFlow." http://flowgrammable.org/sdn/openflow/. [Online].
[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, *et al.*, "P4: Programming Protocol-independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.
[4] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*, pp. 437–455, Springer, 1990.
[5] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[6] "What is Ethereum?." https://ethereum.org/. [Online].
[7] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, pp. 1–37, 2014.
[8] A. Rosic, "Smart contracts: The blockchain technology that will replace lawyers," *Retrieved from Blockgeeks: https://blockgeeks.com/guides/smart-contracts*, 2017.
[9] N. L. Van Adrichem, B. J. Van Asten, and F. A. Kuipers, "Fast recovery in software-defined networks," in *2014 Third European Workshop on Software Defined Networks*, pp. 61–66, IEEE, 2014.
[10] C.-Y. Chu, K. Xi, M. Luo, and H. J. Chao, "Congestion-aware single link failure recovery in hybrid sdn networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1086–1094, IEEE, 2015.
[11] A. Capone, C. Cascone, A. Q. Nguyen, and B. Sanso, "Detour planning for fast and reliable failure recovery in sdn with openstate," in *2015 11th international conference on the design of reliable communication networks (DRCN)*, pp. 25–32, IEEE, 2015.
[12] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 78–85, 2017.
[13] S. R. Basnet and S. Shakya, "BSS: Blockchain security over software defined network," in *International Conference on Computing, Communication and Automation (ICCCA)*, pp. 720–725, IEEE, 2017.
[14] Z. A. El Houda, L. Khoukhi, and A. Hafid, "Chainsecure-a scalable and proactive solution for protecting blockchain applications using sdn," in *Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.
[15] T. Alharbi, "Deployment of blockchain technology in software defined networks: A survey," *IEEE Access*, 2020. Available at: http://doi.org/10.1109/access.2020.2964751.
[16] Y. Liang, H. Yang, Q. Yao, S. Guo, A. Yu, and J. Zhang, "Blockchain-based efficient recovery for secure distributed control in software defined optical networks," in *Optical Fiber Communication Conference*, pp. Th1G–1, Optical Society of America, 2019.
[17] Z. Noshad, A. Javaid, M. Zahid, I. Ali, N. Javaid, *et al.*, "Node recovery in wireless sensor networks via blockchain," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 94–105, Springer, 2019.
[18] M. Kuźniar, P. Perešíni, N. Vasić, M. Canini, and D. Kostić, "Automatic failure recovery for software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 159–160, ACM, 2013.
[19] P. M. Mohan, T. Truong-Huu, and M. Gurusamy, "TCAM-aware local rerouting for fast and efficient failure recovery in software defined networks," in *Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2015.
[20] K.-C. Fang, K. Wang, and J.-H. Wang, "A fast and load-aware controller failover mechanism for software-defined networks," in *International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pp. 1–6, IEEE, 2016.