# Fault Tolerant SDN Controller: A Survey

S.Sujitha[#1], K.Pallavi Priya[*2], B.Pragathi[*3]

[#]Assistant Professor, Department of Information Technology,

*UG Scholar, Department of Information Technology

Thiagarajar College of Engineering, Madurai, Tamil Nadu, India.

***Abstract*—Software-Defined Networks is an emerging technology for managing, controlling and separating the network's control logic from the underlying routers and switches, and introducing the ability to program network operations. The Controller is the core of SDN architecture, which performs the operations of a network administrator. SDN Controller is subject to Single Point of Failure; whereas there is a fault in controller it will affect the whole network performance. To enhance security in SDN, innovative security services and applications that are to be built upon SDN capabilities. In this position paper we argue for the need to build secure, fault tolerant and dependable SDN controller by design. In particular, we address the design of fault tolerant SDN controller – with a focus on aspects such as availability, performance, security and dependability. We undertake a comprehensive survey of recent works that apply to fault tolerant SDN controller and identify promising future directions that can be addressed by such research and it also surveys latest developments in this active research area of SDN.**

Keywords: SDN Controller, Fault Tolerance, Availability, Replication.

## I. INTRODUCTION

Software-Defined Networking (SDN) [5, 6] has emerged as the network architecture where the control plane logic is decoupled from the forwarding plane. It is a new approach for network programmability, which refers to the ability to control, change, and manage network behavior dynamically through software via open interfaces and proprietary defined interfaces. The SDN framework enhances centralized control of data path elements independently of the network technology used to connect the network devices. The centralized control embeds all the intelligence and maintains a network-wide view of the data path elements and the network links. This centralized view makes the controller suitable to perform network administration functions while allowing easy modifications to the networking functions through the centralized control plane. SDN provides reusability, because a single high level program can be implemented for multiple data-transfers. It provides rapid innovation by eliminating the dependence of hardware embedded services and it also uses multiple controllers to provide reliability, availability and handle the traffic-load in the network.

## II. SDN ARCHITECTURE

SDN architecture consists of three planes namely, Application, Control and Data [7]. These different planes are connected by different CPI's (Control Plane Interface).

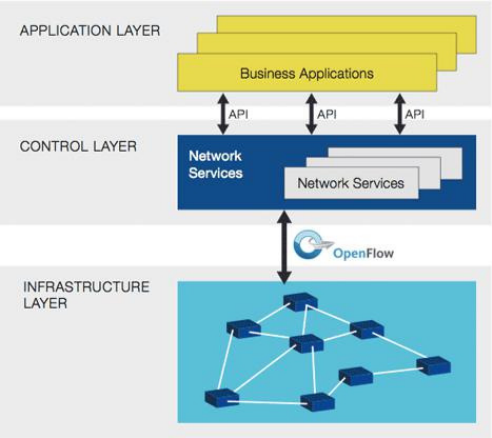


Fig1. **SDN Architecture**

Application plane and Control plane are connected using North bound APIs and it contains applications like security system, IDS, monitoring services, etc., and some other general applications

which require access to network devices. Applications connect to SDN controller agent via A-CPI, provided by controller agent. Then applications demand service of those resources from SDN controller to transfer data. Control Plane and Data Plane are connected through South bound APIs, SDN controller then decides the efficient logic for traffic flow and sends the control to data plane for actual transfer of data and acknowledges the application. The control plane is core element of SDN Architecture. It contains SDN controller which provides centralized control and it acts as a network administrator. All programming logic about packet forwarding, all network switching decisions and network routing are programmed dynamically inside SDN-Controller. This high level program is then transferred to data plane. It is the part that manipulates forwarding devices through a controller to achieve the specific goal of the target application. The high level control structures are transferred from control plane to data plane using secure transfer protocols like Open Flow, most widely used SDN-Controller protocol for secure control flow. Due to all these beneficial factors, several cloud service providers and big data centres are looking forward to SDN. We need to address scalability, availability, and resilience when building SDNs.

## III. SECURITYTHREATS AND VULNERABILITIES

SDN can significantly improve network applicability and efficiency; it is exposed to new threats that are more serious than those in traditional networks. The various categories of threats vectors [8] and attacks associated with SDN layered framework is defined in SDN architecture are the following,

1. Forged or Faked traffic flows
2. Attack on and vulnerabilities in switches
3. Attacks on control plane communications with Northbound and Southbound APIs
4. Attacks on and susceptibilities in controllers
5. Lack of mechanisms to ensure trust between the controller and management applications
6. Attacks on and weaknesses in administrative stations
7. Lack of confidential resources for forensics and remediation

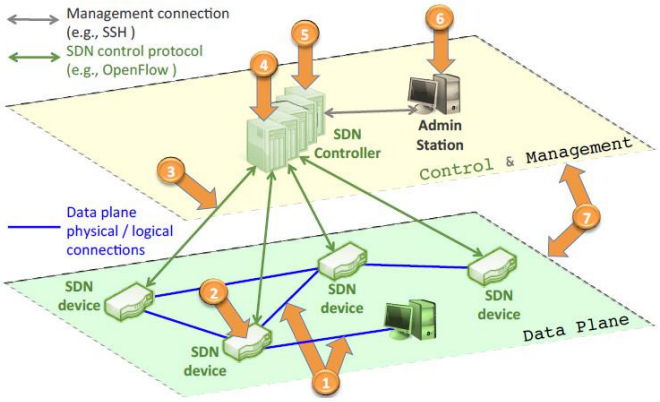


Fig2. **Security Threats**

The seven threat vectors includes information about it's to SDN. Threat vectors 3, 4, and 5 are specific to SDNs and are not present in traditional networks. These threat vectors arise from the separation of the control and data planes i.e the logically centralized controller. The attacks on and vulnerabilities in controllers is the severe threats to SDN. Replication is one of the most possible solutions for it to improve the dependability of the system. It would detect, remove or mask abnormal behaviour of SDN Controller.

There are different security attacks like Data leakage, IP spoofing, unauthorized access, data modification, denial-of-service, malicious applications that are possible at different parts of SDN framework.

Fault tolerant Controller is an essential part of SDN, and this property should be addressed while designing SDN architecture.

SDN fault tolerance covers different fault domain [9]:

The Data Plane (Switch or link failure)

The Control Plane (failure of switch-controller controller)

The Controller itself.

In the past years we are seeing a steady increase in the number of SDN-based applications being deployed in production networks. Google has deployed SDN architecture to connect its data centers across the world. This network has been in deployment for 3 years, with success, and helps the company to improve operational efficiency and reduce costs significantly. In this paper, we survey on the importance of SDN controller architecture which is distributed, fault-tolerant, and strongly consistent. The key element of this architecture is a

data store that keeps relevant network and applications state, proving that SDN applications function on a consistent network view, which ensures coordinated and correct behaviour, and consequently simplified application design.

## IV. RELATED SURVEY

### CORONET: Fault Tolerance for Software Defined Networks

The goal of this work [13] is to develop a fault tolerant SDN architecture that can rapidly recover from faults and gradation to larger sized network. This paper presents CORONET, an SDNfault-tolerant system that recovers from multiple link failures in the data plane. CORONET, COntroller based RObust NETwork, is a scalable and efficient fault tolerant system. It has the following properties:

- Fast recovery
- Scalable to large networks
- Multipath routing
- Works with arbitrary networks
- Single control plane

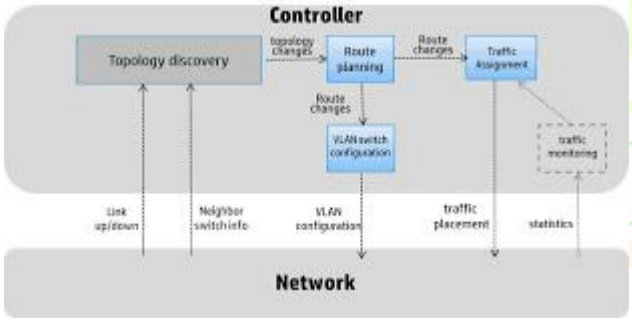


Fig3. **CORONET Architecture.**

CORONET's fig 3 use of VLAN dramatically simplifies packet forwarding, reducing the number of forwarding rules and thus improving scalability compared with a standard Open flow approach. SDN applications in CORONET can only specify logical paths implemented by VLANs. While many existing Open flow applications directly control the packets and these applications could be rewritten using the CORONET framework. The Topology discovery module periodically collects topology information and receives asynchronous events upon link/switch failure. It ensures that CORONET has up-to-date information about the network status. The Route planning module calculates multiple routing paths based on the topology information. VLAN growing algorithm computes growing paths, which creates multiple link disjoint shortest routing paths using Dijkstra's shortest path algorithm. When a link fails the link-disjoint property provides a better reliability by minimizing the number of affected routing paths. The VLAN switch configuration module configures multiple switch ports and the Traffic assignment module assigns host traffic to routing paths. Currently, the algorithm assigns host traffic to a routing path (VLAN ID) in a randomly fashion. However, we envisage to incorporate a separate traffic monitoring module (as shown as a dotted box in Figure 3), so that the module can perform dynamic load balancing.

The CORONET controller is built on top of NOX [14], a platform that provides APIs for SDN applications which is used to interact with Open Flow switches. CORONET is consistent with Open Flow specification version 1.0.0 [15]. We evaluate our CORONET prototype using a virtual network topology emulator called Mininet [16], which is used to generate customized virtual network topologies in a Linux machine. Thus CORONET only supports fault-tolerance for data plane failures. It simplifies packet forwarding and thus improving scalability.

### On the feasibility of a consistent and fault-tolerant data store for SDNs

In this systems techniques [17] it is possible to build a strong consistent, fault-tolerant SDN controller framework which achieves a better performance. The core element in this architecture is too replicate controller which provides a highly-available, strongly consistent data store. By using the state of-the-art replication algorithm, we integrate the Floodlight controller with a data store in the distributed controller architecture.

The key idea of our controller fig 4. Architecture is to make the controller instances coordinate with the data store where the relevant state of the network is maintained in a consistent way. To avoid any single point of failure, the data store is implemented with more no of servers (replicas), without affecting the consistency. State machine replication (SMR) is one of the most

popular techniques for implementing replicated data store and it uses Paxos algorithm which ensures that all updates in the data store are applied to all the replicas in the same order.

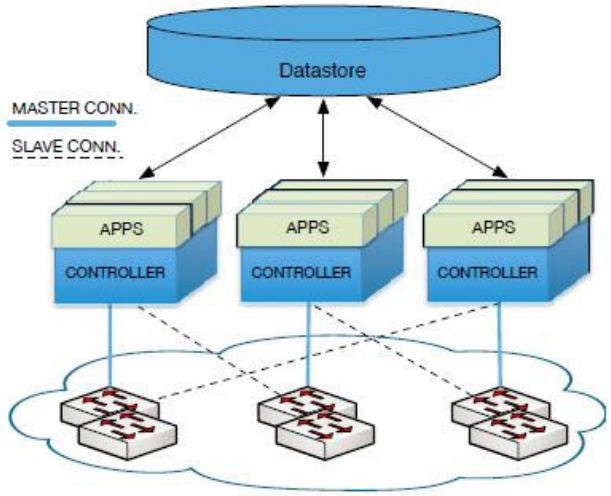


Fig4.**THE SHARED DATA STORE CONTROLLER architecture**

The architecture consists of a set of SDN controllers connected to the switches in the network. All decisions of the control plane are based on Open Flow events triggered by the switches and the data store backup controllers keep monitoring this primary, as in the distributed controller design. If the primary fails, one of the backups which has the highest IP address takes the role of primary and it inform to other controllers and it also uses the data store for controlling the network. It can deal with faults in the control plane (the connection controller-switch) by having each switch connected to several controllers (which is ongoing work). In this paper they proposed a distributed, highly available, strongly consistent controller for SDNs

**BYZANTINE-RESILIENT Secure Software-Defined Networks with Multiple Controllers in Cloud**

The centralized control plane introduced by SDN [18] imposes a great challenge for the network security. In this paper, we present a secure SDN structure, it employs multiple controllers for each switch [19] and it is also managed by other devices, not just as a traditional manner. When a controller is disabled because of attacks, a backup one can be immediately activated to take over the controlling function. We design SDN architecture to resist the attack on the control plane by BFT mechanism in Cloud. In traditional SDN architecture, each switch is controlled by a single controller [20].Now, we propose to use multiple controllers link with other fig 5 for confirming the update of flow tables in each switch. First, each switch requires different number of controllers, and each controller provides services to multiple switches. Second, as the

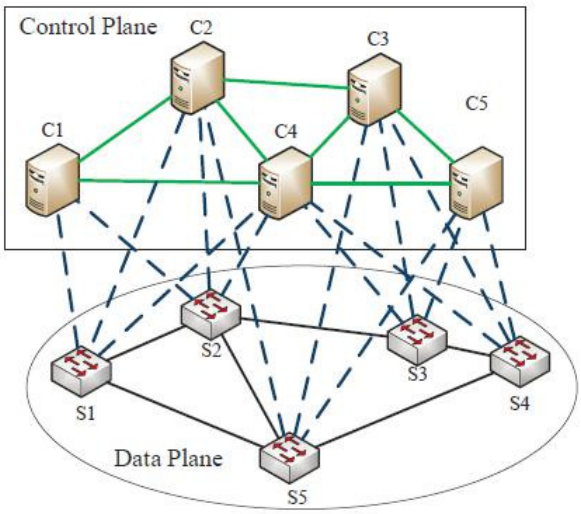


Fig5**. BYZANTINE architecture**

Services are replicated and executed on independent replicas. One replica act as the primary in a view where the others are secondary (backups). When the primary fails, the view will change. For this they propose an algorithm, called requirement first assignment (RQFA) algorithm, for solving the CAFTS (controller assignment in fault-tolerant SDN) problem. The algorithm works briefly as follows.

1) The client sends a request to the primary
2) The primary sends the request to other replicas.
3) Replicas execute the request and it sends a reply to the client.
4) The client waits for the reply from different replicas with same result.

From the result, the architecture has little impact on the network latency, and it provides better security than general distributed controller. The proposed algorithm performs higher efficiency

than random assignment and to minimize the number of controllers while satisfying the security requirements of a given set of switches.

**SMARTLIGHT: A Practical Fault-Tolerant SDN Controller**

In this paper, [10] they present a design of a fault-tolerant controller, and materialize it by proposing the architecture for small to medium-sized networks. The proposed design guarantees a smooth transition in case of failures and avoids the need of an additional coordination service. To ensure the network is controlled despite faults in this controller, it is deployed with several controller replicas. All switches therefore establish a connection with all controllers. Several controllers are deployed. Fig6. A single one in the network act as primary, whereas the others are used as backups. If the primary controller is crashed then the other controllers can take over the role of primary by using leader election. To enable this fault-tolerant solution, we have algorithms for fault detection and leader election [11].This ensures a smooth transition to a new primary.

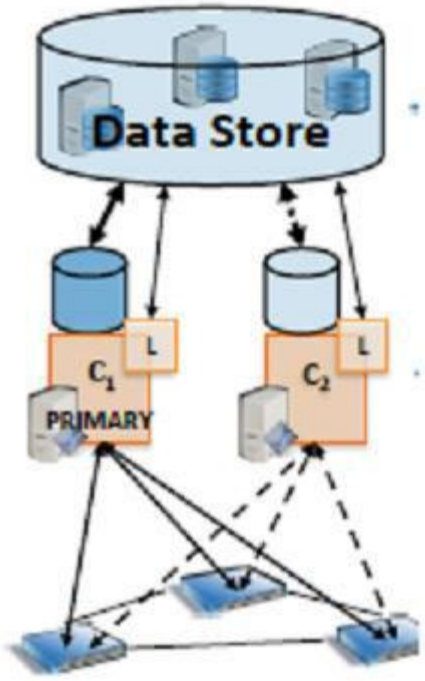


Fig6.**SMARTLIGHTArchitecture.**

When the primary fails, the new primary takes over the role of the old primary and its first action is to read the current state from the data store .Because the network state in the data store is always up-to-date then the new primary will have a consistent view of the network and it ensures that it has no single point of failure .A possible solution-Paxos algorithm [12] for implementing the data store as a Replicated State Machine (RSM) [20]. It gives us the guarantees that a data store update will be reflected in successive reads. It also ensures that no update performed by the primary on a data store will be lost after a failure Implementation of a fault-tolerant controller – SMaRtLight. The SMaRtLight architecture is presented in Fig6.

It includes additional aspects to implement the data store in the coordination service. The controllers maintain a local cache to avoid accessing the shared data store. There is only one active primary controller are used for accessing the data store at any one time, because the cache does not require synchronization technique. Only the primary controller communicates with the data store, reading from or writing to the application-required tables. Simultaneously, it also updates its cache.

## V. CONCLUSION AND FUTUREWORK

SDN is considered as a promising solution to meet the demands like, more convenient Internet access, more bandwidth from users, and also more dynamic management from service providers. SDN will obtain more appropriate control of the infrastructure to achieve more efficient infrastructure resource utilization. In this paper, we have surveyed a wide range of recent and state-of-the-art projects in fault tolerant SDN Controller. Moreover, we have provided a literature survey of recent SDN researches in the control layer. As future work, we will focus on the optimization of the proposed distributed controller and on modifying the Floodlight applications to make them "data store-aware". As the number of SDN production networks increase the need for dependability becomes essential. The key takeover of this work is that dependability mechanisms have their cost, and it is therefore an interesting challenge for the SDN community to integrate these mechanisms into scalable control platforms.

## REFERENCES


[1] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI'09, Berkeley, CA, USA, 2009.

[2] Press Release. "Hacking Habits" Survey Cites Misconfigured Networks As The Main Cause Of Breaches. Tufin Technologies, 31 August, 2010. http://www.tufin.com/about-us/news-and-media/pressreleases/ 2010/august-31,-2010/.

[3] R. J. Colville and G. Spafford. Configuration Management for Virtual and Cloud Infrastructures. Gartner Inc., 27 October, 2010. http://www.gartner.com/id=1458131.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson. Rexford, S. Shenker, and J. Turner. Open Flow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer Communication Review (CCR), 38(2):69–74, 2008

[5] N. McKeon, "How SDN will Shape Networking," October 2011.[Online]. Available: http://www.youtube.com/watch?v=c9-K5O qYgA

[6] S. Schenker, "The Future of Networking, and the Past of Protocols, "October 2011. [Online]. Available: http://www.youtube.com/watch?v=YHeyuD89n1Y

[7] W. Xia, Y. Wen, C.H. Foh, D. Niyato, H. Xie, A survey on software defined networking, Commun. Surv. Tutorials, IEEE PP (99) (2014).

[8]Towards Secure and Dependable Software-Defined Networks, Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo University of Lisbon, Portugal, August 16, 2013.

[9] European Workshop on Software Defined Networks.[Online].Available: http://www.ewsdn.eu/previous/ewsdn12.html

[10] SMaRtLight: A Practical Fault-Tolerant SDN Controller Fabio Botelho Alysson Bessani Fernando M. V. Ramos Paulo Ferreira LaSIGE/FCUL, University of Lisbon, Portugal.

[11] P. Hunt et al. Zookeeper: Wait-free coordination for Internet-scale services. In USENIX ATC, 2010.

[12] L. Lamport. The part-time parliament. ACM Trans. Computer Systems, 16(2):133–169, May 1998.

[13] CORONET: Fault tolerance for software defined networks. In IEEE ICNP, 2012. F. Botelho, F. Ramos, and A. Bessani.

[14] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: towards an operating system for networks. *ACM,* July 2008.

[15] Open Flow Specification v1.0. http://www.openflowswitch.org/documents/openflow-spec-v1.0.0.pdf

[16] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks (at scale!). In Proc. HotNets, Oct. 2010.

[17] On the feasibility of a consistent and fault-tolerant data store for SDNs. In EWSDN, Oct. 2013. F. Botelho, A. Bessani, and F. Ramos.

[18] Byzantine-Resilient Secure Software-Defined Networks with Multiple Controllers in Cloud.He Li, Peng Li, Song Guoand Amiya Nayak, IEEE, Oct 2013.

[19] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "Devoflow: Cost-effective flow management for high performance enterprise networks," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, ser. Hotnets-IX. New York, NY, USA: ACM, 2010

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson,J. Rexford, S. Shenker, and J. Turner, "Open flow: enabling innovation in campus networks," SIGCOMM Comput. Commun.Rev. vol. 38, no. 2, pp. 69–74, Mar. 2008.

[21] Floodlight Controller. http://goo.gl/ZlLXdO.

[22] Z. Cai, A. L. Cox, and T. E. N. Maestro, "A system for scalable open flow control," Technical Report TR10-08, Rice University, Tech. Rep., 2010.