Coordination of Controllers and Switches in Software Defined Networks (SDN) for Multiple Controllers

Stavroula Lalou
Department of Digital Systems
University of Piraeus
Piraeus, Greece
slalou@unipi.gr

Georgios Spathoulas
Dept. of Inform. Sec. and Comm. Techn.
NTNU
Gjøvik, Norway
georgios.spathoulas@ntnu.no

Sokratis Katsikas
Dept. of Inform. Sec. and Comm. Techn.
NTNU
Gjøvik, Norway
sokratis.katsikas@ntnu.no

Authors

Stavroula Lalou holds a Diploma of Cultural Technology and Communication from Aegean University of Mytilene since 2007, a MSc in Computer Science from University of Staffordshire of UK since 2011 and she is a full time PhD Student Department of Digital Systems of University of Piraeus since December of 2016.

Dr. Georgios Spathoulas holds a Diploma of Electrical and Computer Engineering from Aristotle University of Thessaloniki since 2002, a MSc in Computer Science from University of Edinburgh since 2005 and a PhD from the Department of Digital Systems of University of Piraeus since 2013. He is a member of Laboratory Teaching Staff of the Department of Computer Science and Biomedical Informatics of University of Thessaly since 2014 and he teaches in both undergraduate and postgraduate study programs of the Department. He is also collaborating, as a post doctoral researcher, with the Critical Infrastructures Security and Resilience group in NTNU CCIS. His research interests are related to network security, privacy preserving techniques and blockchain technology. He is the co-author of more than 40 publications in peer reviewed journals and conference proceedings. He has also served as Program Committee member in international conferences and he has taken part in both national and international research programs.

Sokratis K. Katsikas is the Director of the Norwegian Center for Cybersecurity in Critical Sectors and Professor with the Department of Information Security and Communication Technology, Norwegian University of Science and Technology. He is also Professor Emeritus of the Department of Digital Systems, University of Piraeus, Greece. He received the PhD in Computer Engineering from the University of Patras, Greece, the MSc in Electrical and Computer Engineering from the University of Massachusetts at Amherst, USA, and the Dipl. Eng. Degree in Electrical Engineering from the University of Patras, Greece. In 2019 he has awarded a Doctorate Honoris Causa by the Dept. of Production and Management Engineering of the Democritus University of Thrace, Greece. In 2021 he was ranked 7th in the security professionals category of the IFSEC Global influencers in security and fire list. In the past, among others, he has been the Rector of the Open University of Cyprus; Rector and Vice Rector of the University of the Aegean, Greece; General Secretary of Telecommunications and Posts of the Hellenic Government; Chair of the National Council of Education of Greece; member of the Board of the Hellenic Authority for the Security and Privacy of Communications; and member of the Board of the Hellenic Authority for the Quality and Accreditation of Higher Education. He has authored or co-authored more than 300 journal papers, book chapters and conference proceedings papers. He is serving on the editorial board of several scientific journals, he has co-authored/edited 46 books and conference proceedings, and he has served on/chaired the technical programme committee of more than 800 international scientific conferences. He chairs the steering committee of the ESORICS conferences and he is the Editor-in-Chief of the International Journal of Information Security (Springer).

Aims and Contributions of our paper

- Coordination of the workload among distributed SDN controllers is critical role for both the network performance and the control plane scalability. Therefore, various load balancing techniques were proposed for SDN to efficiently utilize the control plane's resources. However, such techniques suffer from increased latency and packet loss that come as the result of load migration requirements and intensive communication between the SDN controllers. The proposed system adopts OpenFlow mechanism and introduces a new system that offers coordination, synchronization and stable performance.
- In this paper we propose a scalable and crash- tolerant load balancing based on controller switch connection for multiple OpenFlow controllers.

The contribution of this paper is:

- A dynamic coordination and synchronization system among SDN controllers and switches that focus partic-ularly on the impact of the rate of synchronization on the performance of network.
- A system that can dynamically shift the load across multiple controllers through switches.
- A controller fail-over without switch disconnection avoiding the single point of failure problem

OpenFlow Protocol

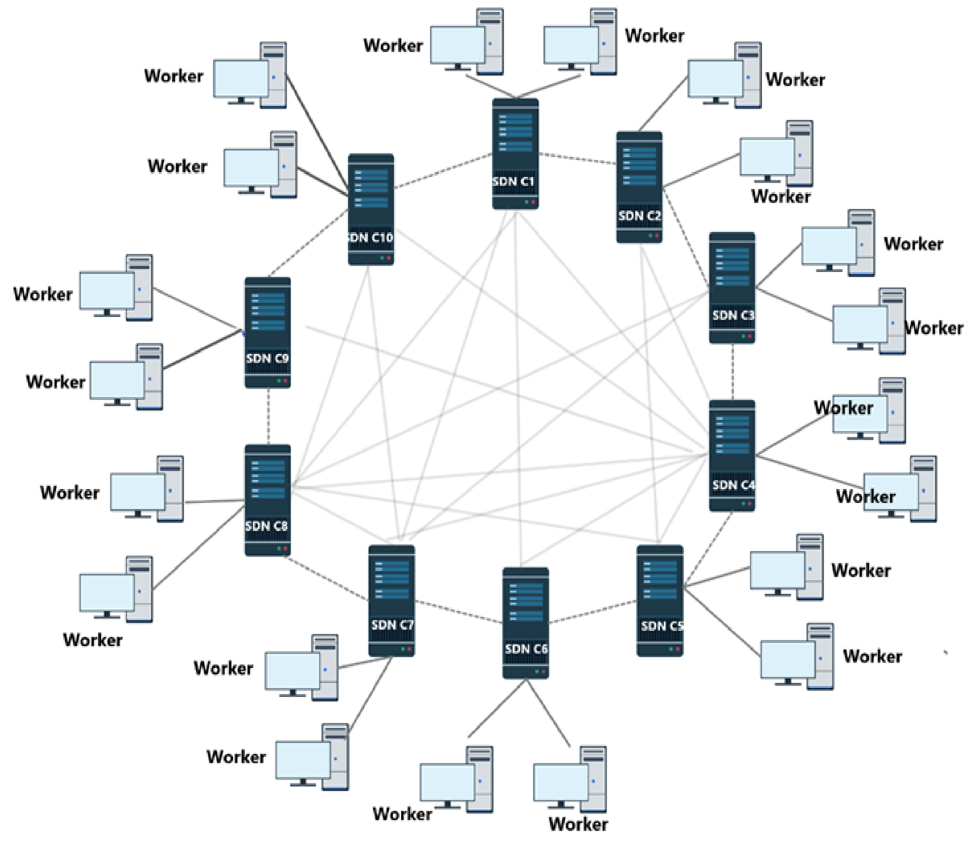
- The OpenFlow architecture consists of numerous pieces of OpenFlow-enabled switching equipment which are managed by one or more OpenFlow controllers.
- An OpenFlow switch contains multiple flow and group tables. Each flow table consists of many flow entries. These are specific to a particular flow and are used to perform packet look-up and forwarding.
- An OpenFlow switch contains multiple flow and group tables. Each flow table consists of many flow entries. The flow entries can be manipulated as desired through OpenFlow messages exchanged between the switch and the controller on a secure channel. By maintaining a flow table, the switch can make forwarding decisions for incoming packets by a simple look-up on its flow- table entries.
- Open-Flow switches perform an exact match check on specific fields of the incoming packets. For every incoming packet, the switch goes through its flow table to find a matching entry. The flow tables are sequentially numbered. The packet- processing pipeline always starts at the first flow table. The packet is first matched against the entries of a flow table. If the packet matches a flow entry in a flow table, the corresponding instruction set is executed. Instructions associated with each flow entry describe packet forwarding, packet modification, group table processing, and pipeline processing.

Openflow Connection Strategy

- The first issue is to address the switch-to-controller connection strategy and how switches are connected to SDN controllers. In early OpenFlow version, switches can only attach to one controller. Furthermore, that link is static, meaning that operators have to configure the switch manually when it needs to attach to a new controller.
- A distributed SDN controllers setup, on the other hand, requires a dynamic connection between switches to controllers. The dynamic connection enables to move a switch from one controller to another controller during a fail-over or load balancing process. Fortunately, there are two options to deploy such flexible switch to controller connection, using the IP alias connection or OpenFlow Master/Slave connection.
- In the Master state, the controller has full access to the switch as in the Equal role. When the controller changes its role to Master, the switch changes the other controller in the Master role to have the Slave role. The role change does not affect controllers with the Equal role. The controller receives from switch asynchronous Port-status messages. The controller can send Asynchronous- Configuration messages to set the asynchronous message types it wants to receive. An OpenFlow instance can connect to one or more controllers, depending on the controller connection mode the OpenFlow instance uses ether Single instance in which the OpenFlow instance connects to only one controller at a time. When communication with the current controller fails, the OpenFlow instance uses another controller, or the Multiple instances so it can simultaneously connect to multiple controllers. When communication with any controller fails, the OpenFlow in- stance attempts to reconnect to the controller after a reconnection interval.

Our Proposal

- The proposed system implements a novel network of multiple controllers using RAFT consensus algorithm to maintain stability, scalability, and consistency, it was presented in a prior work [18].
- In this paper we extend our approach using features from Open-Flow connection methods. It is based in Master/ Slave connection between controllers and switches. It supports the connection and coordination of multiple dis- tributed SDN controllers to serve as backup controllers in case of a failure. According to our experiments the load conditions of controllers, our proposed method can dynamically shift the load across the multiple controllers. Moreover, multiple controllers allow data load sharing when a single controller is overwhelmed with numerous flow requests. In general, our approach can reduce latency, increase scalability, and fault tolerance, and provide enhanced availability in SDN deployments.



Our Proposal

- The proposed mechanism, consists of multiple SDN con- trollers that collaborate to manage the network. Each controller is responsible for a subset of switches in the network. Controllers communicate with each other using a coordination and synchronization mechanism. Controllers exchange their load information with other controllers in the network. We use a consensus-based coordination and synchronization mechanism. Each controller registers with the coordination service and participates in the distributed coordination protocol. The coordination service maintains a shared state, such as network topology information and controller assignments.
- Controllers periodically synchronize their local state with the shared state in the coordination service. This synchronization is achieved by employing a combination of data replication, using flow tables. When a controller joins or leaves the system, the coordination service notifies other controllers to update their view of the network and redistribute the load if necessary.
- Load balancing across controllers can be achieved through dynamic redistribution of switches and their associated flows. Controllers use load balancing algorithms based on factors like controller workload, switch capacity, and network traffic patterns. When load balancing decisions are made, controllers negotiate and transfer the ownership of switches and their flows based on the new load distribution. To handle controller failures, a fail-over mechanism is necessary. When a controller fails, the coordination service detects the failure and triggers a fail-over process. The fail-over process involves selecting a new controller to take over the responsibilities of the failed controller. The new controller establishes connections with the switches managed by the failed controller, ensuring a seamless transition without disrupting network operations.
- Controllers use a standard protocol, the OpenFlow, to communicate with the switches and exchange network control messages. The coordination and synchronization mechanism discussed above enables controllers to exchange coordination messages to maintain consistency and distribute control responsibilities.

Load Balancing

- Controllers exchange their load information with other controllers in the network. The Controller Election Process is been implemented in the controller election process using the OpenFlow protocol. The controllers negotiate and decide which controller will be the master and which will be the backup using protocols, such as OpenFlow's Role Request message.
- Each controller monitors its own load using metrics (CPU utilization, memory usage, or the number of active flows Each controller compares its load metric with the load metrics received from other controllers. The comparison helps identify the least loaded controller among the available options. If the controller determines that it is the least loaded based on the load comparison, it continues to handle incoming traffic as usual. If the controller determines that another controller has a lower load, it takes appropriate actions for load balancing. The load balancing decisions can be implemented by modifying the flow table entries in the switches, redirecting traffic to the appropriate controllers based on the load balancing algorithm. The SDN controllers use the OpenFlow protocol to install, update, or remove flow rules dynamically to achieve load balancing.
- When a new request arrives at a switch, the switch forwards the request to its designated controller. The OpenFlow protocol allows switches to direct incoming packets to a specific controller based on rules defined in the flow tables. Configure the flow tables in the switches to match and forward the incoming requests to the appropriate controller based on load balancing policies.
- Each switch maintains the load information received from the controllers it is connected to. The switch compares the load information of the connected controllers. Based on the comparison, the switch selects the least loaded controller as the destination for incoming requests.By leveraging the capabilities of the OpenFlow protocol, the switch can make informed decisions about which controller to forward incoming requests to, ensuring load balancing among the controllers in the SDN system.
- To implement load balancing, packet fields, such as source IP address, destination IP address, transport protocol are record in the flow table entries to direct packets to the desired controller. We use the OpenFlow protocol to set the flow action in the flow rules of the switches.

Performance evaluation- Experimental Setup

To evaluate the performance of the proposed mechanism we conducted a network simulation. We evaluated the performance of our system in terms of load balancing in terms of response time, throughput, packet lost, delay and the time overhead imposed by the controllers and switches to coordinate.

The coordination performance and scalability between controllers, switches and hosts also have been depicted according to the scenario of routing several packets that are successfully routed (without traversing any failed link) to their destinations. We emulate the performance using Mininet and Ryu [17] component-based software defined networking framework. Ryu [17] provides software components with well- defined API that make it easy for developers to create new network management and control applications. and created a topology of 10 SDN controllers, consisting of one master controller and nine SDN controllers, along with 20 switches.

- Master Controller: Controller M
- SDN Controllers: Controller C1, Controller C2, Con-
- troller C3, Controller C4, Controller C5, Controller C6,
- Controller C7, Controller C8, Controller C9
- Switches 1-20: S1, S2, S3, S4, S5, S6, S7, S8, S9, S10,
- S11, S12, S13, S14, S15, S16, S17, S18, S19, S20

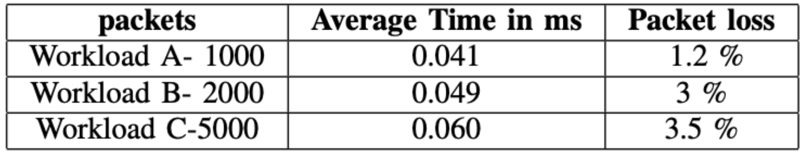
Master Controller M does not handle any switches directly. Each controller can handle up to three switches. We have assigned an initial load distribution of switches to controllers. Controller C1: S1, S2, S3; Controller C2: S4, S5, S6; and so on. We have set initial metric values for each controller (CPU utilization, memory usage, number of active flows) based on the simulation scenarios. Periodically collect metrics from each controller and switch and update the metric values based on the simulated workload and network conditions.

Performance evaluation- Experimental Setup

- Calculates a score for each controller based on the weighted metrics. Then identifies the controller with the highest score as the ”over loaded” controller and the one with the lowest score as the ”under loaded” controller.
- Defines a threshold value to determine when a controller is considered overloaded or under loaded. If the score difference between the overloaded and under loaded controllers exceeds the threshold, initiate load redistribution. Determines a subset of switches to be transferred from the overloaded controller to the underloaded controller.
- For example, if Controller C1 is overloaded and Controller C2 is under loaded, it can transfer S1 and S2 from Controller C1 to Controller C2. It updates the flow tables of the affected switches to redirect traffic to the under loaded controller. The under loaded controller assumes control of the transferred switches and their associated flows. According to the Master/slave constraint the switch can be controlled by more than one SDN controllers but only one master controller at time
- Each controller is connected to the other controllers and the available switches. Each controller can handle up to 3 switches, which are used as the traffic generator to initiate UDP flows to any other host in the network. The performance of routing application is determined by the number of packets that are successfully routed to their destinations.

- We emulate the performance for three different scenarios of workload to test the controllers coordination and the management of the switch. Our system shifts dynamically the load across the switches and the controllers. We simulated three different workloads to stress controllers through adjusting the flow rate. For the first scenario we sent 1000 packets in a time of 100ms, for the second 2000pps and for the third we flood the network to see how it performs and how the nodes coordinate under heavy load.

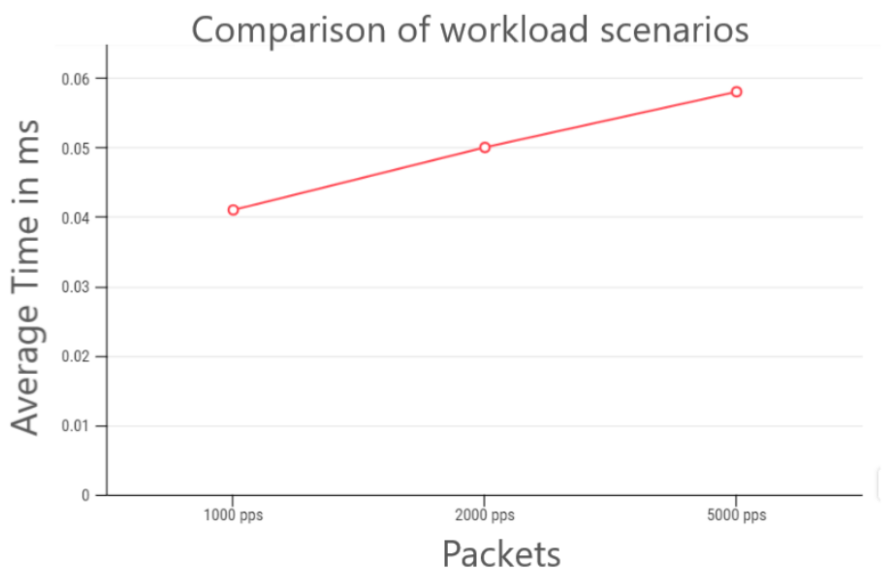
TABLE I
NETWORK PERFORMANCE PACKETS SEND PER SECOND



packets	Average Time in ms	Packet loss
Workload A- 1000	0.041	1.2 %
Workload B- 2000	0.049	3 %
Workload C-5000	0.060	3.5 %

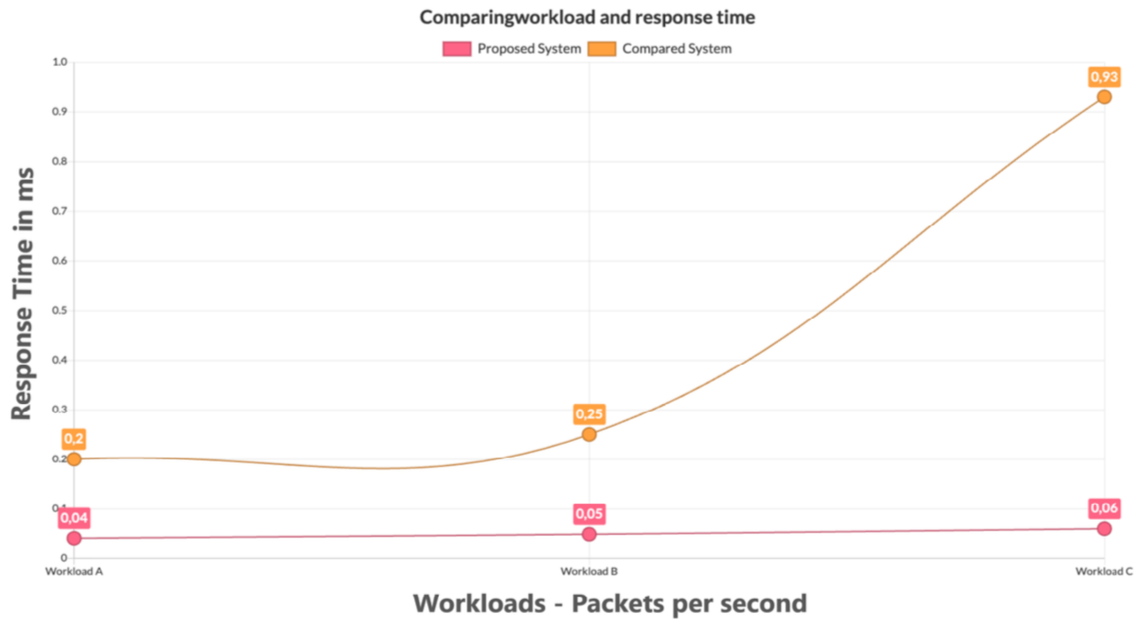
Performance evaluation Results

- We tested the communication between all nodes of the network for a time duration of 100 sec, as in a OpenFlow network the controller response time directly affects the flow completion times. We evaluated the average response time at 0.041 ms, for sending packets throughout the network. The performance of routing application is determined by the number of packets that are successfully routed (without traversing any failed link) to their destinations.
- To analyse the load balancing algorithm, we simulated different network scenarios and workload conditions. Vary the weights assigned to different metrics and observed the resulting load distribution among controllers.
- We emulate the performance for three different scenarios where all the controller synchronize at the same rate equal to (i) 0.041ms (ii)0.049 ms, (iii)0.060 ms, (messages per second).



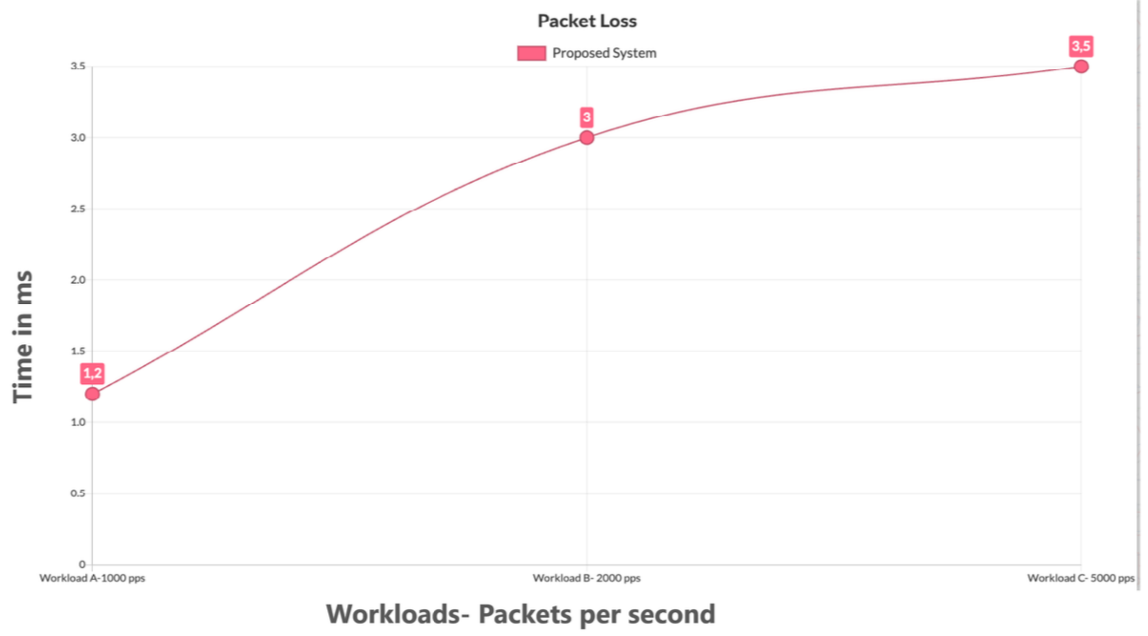
Performance evaluation Results

- We evaluate the response time of the systems by hping command. As Figure shows, the workload significantly affected response time. Comparing the response time in [5], it increases marginally up under workload B and goes up higher under workload C. That is because once the packet interval rate exceeded the capacity of the controller, queuing causes response time to shoot up. Finally, we measure the the time overhead caused by assigning roles to the switches and the cost of switch migration process in the compared system [5]. We observe the migration process takes about 2ms under workload A and increases as the. The failover process takes about an average of 20ms, which mostly affected by the failure detection based on heartbeat messages provided by JGroups. In our proposed system, the average required time for assigning Master role to a controller node is 10,06 ms.



Performance evaluation Results

- We also tested the packet loss. We define the delay to have a normal distribution, which provides a more realistic emulation of networks. As a result, all packets leaving the controller C1 on its interface C1-eth0 will experience delay time which is normally distributed between the range of 10ms ± 20ms, we have consider this delay due to the master election. Also NETEM permits user to specify a distribution that describes how delays vary in the network.
- Usually delays are not uniform, so it may be convenient to use a non-uniform distribution such as normal. For this test, we specified a normal distribution for the delay in the emulated network. In a network, packets may be lost during transmission due to factors such as bit errors and network congestion. The rate of packets that are lost is often measured as a percentage of lost packets with respect to the number of sent packets. The results indicated that there was a small and stable packet loss starting with 1.2% up to 3.5 % and almost all packets were received successfully.



Conclusion

- SDN aims to simplify network architecture and makes it possible to build programmable and agile flexible networks. According to the experimental results that were presented, the proposed system can efficiently coordinate and synchronize the controllers and switches of the network in stable and low time, thus ensuring good performance at all times irrespective of the traffic dynamics. Also, it supports high- throughput, fault- tolerance, and controller synchronization. The result of evaluation showed that our method can improve the communication of all network nodes and improve the throughput and response time of control plane. It can maintain system coordination and network stability and the average response time in all workload tests are low.

References

[1] “OpenNetworkingFoundation.”,https://opennetworking.org(Retrieved July 2023).

[2] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, “Distributed SDN controller system: A survey on design choice,” Computer Networks, vol. 121, no. 5, pp. 100–111, Jul. 2017, doi: https://doi.org/10.1016/j.comnet.2017.04.038.

[3] ”OpenDayLight”, https://www.opendaylight.org (Retrieved July 2023).

[4] P. Berde et al., “ONOS,” Proceedings of the third workshop on Hot topics in software defined networking, Aug. 2014, doi: https://doi.org/10.1145/2620728.2620744.
[5] L. Chu, et al. ”Scalable and Crash-Tolerant Load Balancing Based on Switch Migration for Multiple,” 2014.
[6] OpenFlow Switch Consortium, https://opennetworking.org/?s=openflow (Retrieved July 2023).
[7] [7]A. R. Curtis, et al., “DevoFlow,” ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, pp. 254–265, Oct. 2011, doi: https://doi.org/10.1145/2043164.2018466.
[8] V. Yazici, M. S. ”Controlling a software-defined network via distributed controllers”, 2014.
[9] R. Y. Shtykh and T. Suzuki, “Distributed Data Stream Processing with Onix,” in IEEE International Conference on Big Data and Cloud Computing (BdCloud), IEEE, 2014.

[10] S. Yeganeh and Y. Ganjali, “Kandoo: a framework for efficient and scalable offloading of control applications,” Semantic Scholar, 2012. [11] A. Tootoonchian and Y. Ganjali, ”HyperFlow: a distributed con- trol plane for OpenFlow”, in Proceedings of the 2010 internet net- work management conference on Research on enterprise networking (INM/WREN’10), USENIX Association, USA, 2010.

References

[12]A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, “Towards an elastic distributed SDN controller,” ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, pp. 7–12, Aug. 2013, doi: https://doi.org/10.1145/2534169.2491193.
[13] M. Yu, J. Rexford, et al. ”Scalable Flow-Based Networking with DIFANE,” ACM SIGCOMM Comput. Commun. Rev., pp. 351–362, 2010.
[14] K. Poularakis, et al. Learning the Optimal Synchronization Rates in. arXiv:1901.08936v1 [cs.NI], 2019.
[15] [11]D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, “Logically centralized?,” Proceedings of the first workshop on Hot topics in software defined networks, Aug. 2012, doi: https://doi.org/10.1145/2342441.2342443.

[16]“Iperf.” http://iperf.sourceforge.net (Retrieved July 2023).
[17]“Ryu omponent-based software,”https://ryu-sdn.org/ (Retrieved July 2023).
[18] S. Lalou, et al. “Efficient Consensus Between Multiple Controllers in Software Defined Networks (SDN),” in The Sixteenth International Con- ference on Emerging Security Information, Systems and Technologies, IARIA, pp. 35–40, 2022.