# Efficient Consensus Between Multiple Controllers in Software Defined Networks (SDN)

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

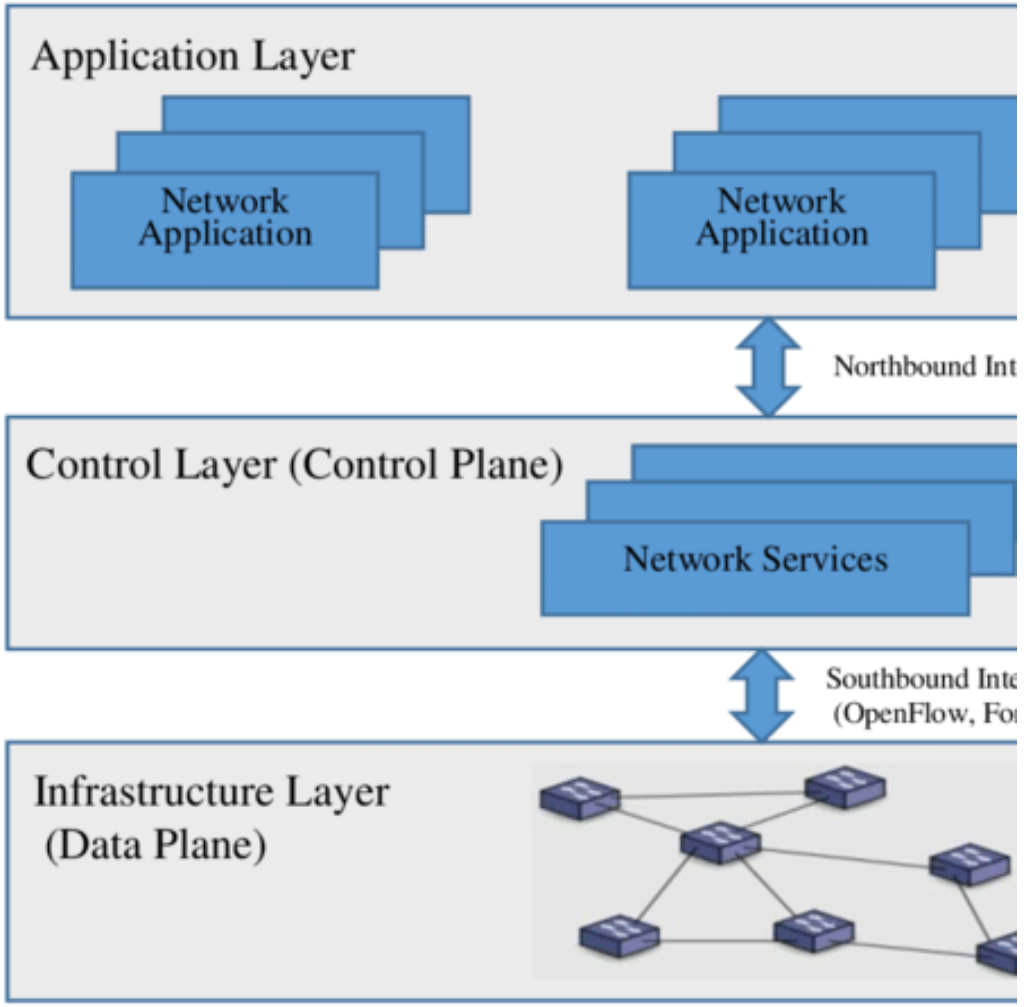
- The use of a single SDN controller offers flexibility and efficiency in network management but leads to problems such as single points of failure and scalability issues. Multiple SDN controller architectures helps to address the above issues. The synchronization of the network state information among controllers is a critical problem, known as the controller consensus problem. To synchronize the network information between controllers, a proper consistency model should be chosen. Strong consistency and eventual consistency are two consistency models commonly used in distributed systems.
- In this paper, we introduce a novel mechanism that supports the operation of multiple controllers in an SDN network.
- The mechanism achieves network flexibility and enhances network management; it also synchronizes the network state between different controllers, while addressing single point of failure, fault tolerance and scalability issues. To demonstrate the practicality of the proposal, we present an implementation with the Raft algorithm for state machine replication, whose performance we evaluated and compared to that of an existing alternative by means of experimentation.

The contribution of this paper is:

- The analysis of the existing mechanisms and protocols for SDN networks
- The definition of the consensus problem for distributed SDN controllers.
- The introduction of a mechanism that supports high throughput, dynamic view changes, fault tolerance, and controller synchronization in multiple SDN controllers setups.

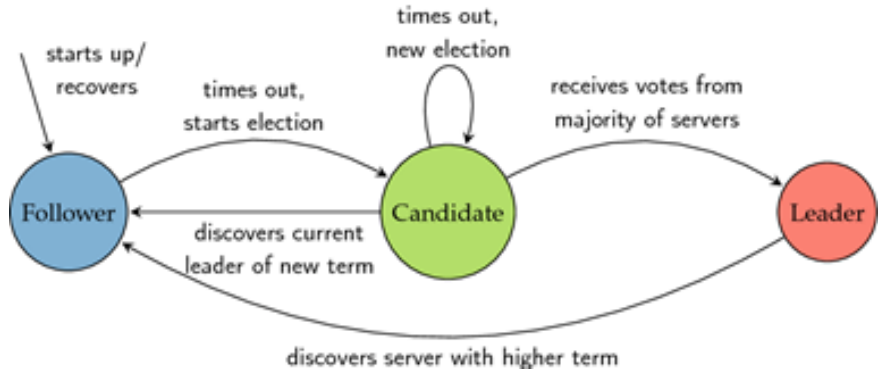
# SDN Architecture

- The control and data planes are decoupled. Control functionality is removed from network devices that will become simple (packet) forwarding elements.
- Forwarding decisions are flow-based, instead of destination-based. A flow is broadly defined by a set of packet field values acting as a match (filter) criterion and a set of actions (instructions). In the SDN/OpenFlow context, a flow is a sequence of packets between a source and a destination..
- Control logic is moved to an external entity, the socalled SDN controller or Network Operating System (NOS). The NOS is a software platform that runs on commodity server technology and provides the essential resources and abstractions to facilitate the programming of forwarding devices based on a logically centralized, abstract network view. It is similar to that of a traditional operating system].
- The network is programmable through software applications running on top of the NOS that interacts with the underlying data plane devices. This is a fundamental characteristic of SDN, considered as its main value proposition.



# Raft Consensus Algorithm

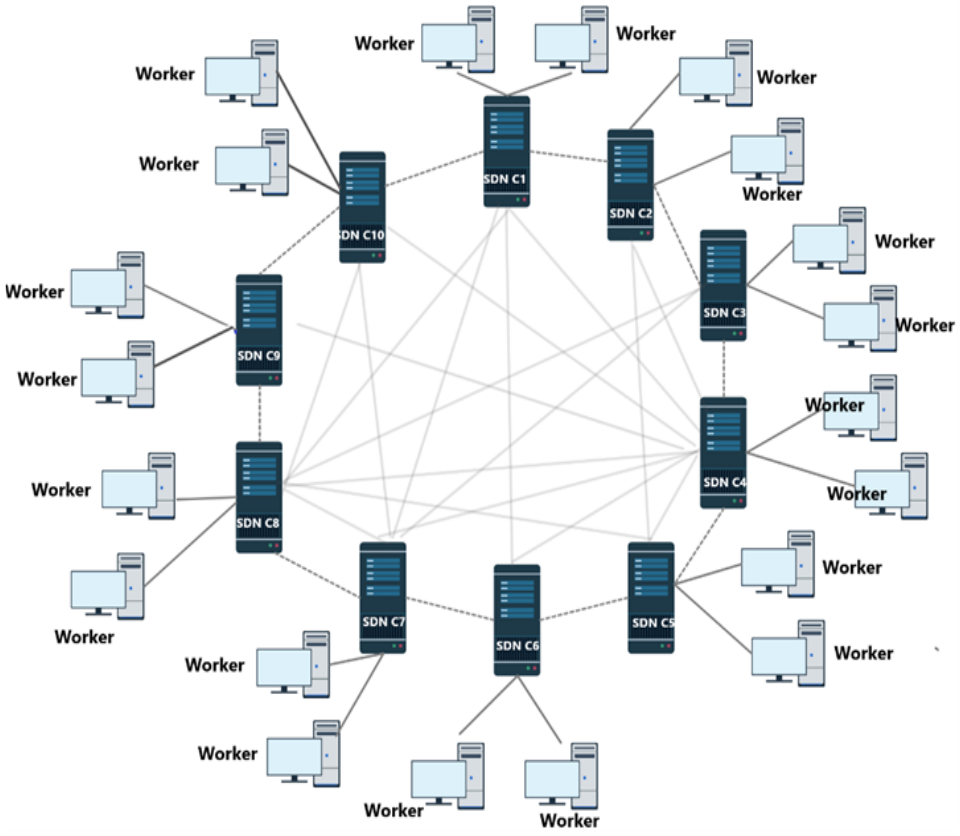
- Raft protocol tries to solve and that is achieving Consensus. Consensus means multiple servers agreeing on same information, something imperative to design fault-tolerant distributed systems.
- In Multiple SDN controllers Raft can ensure scalability and consistency among controllers.
- The Raft algorithm, is a significant consensus algorithm for for state machine replication  and managing a replicated log.
- A leader election algorithm is integrated into the Raft algorithm to ensure consistency.
- Raft separates the key elements of consensus, such as leader election, log replication, and safety, and it enforces a stronger degree of coherency to reduce the number of states that have to be considered in order to reach consensus.
- It also includes a mechanism for changing the cluster membership, which uses overlapping majorities to guarantee safety.
- There are three different node states, namely *leader*, *candidate*, and *follower*.



# Our Proposal

- The proposed mechanism implements a novel network of multiple controllers using the Raft consensus algorithm. It supports the connection and coordination of multiple distributed SDN controllers to serve as backup controllers in case of a failure.
- 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 provides enhanced availability in SDN deployments.
- The proposed mechanism consists of a set of independent controllers (nodes), each one of which stores the required data in its memory. Each controller (node) is assigned with a unique id. In our implementation and test scenario we used a number of nodes in different states.
- Each one can be in one of three different states:

1. Master: In the Master state the node manages and controls the network while it can also process data and send update information to the other controllers.
2. Candidate: the Candidate state the node can send and receive data to/from the other nodes. A Candidate node with the updated data can potentially transit to Master state if the current Master node fails.
3. Worker: Finally in the Worker state the node passively receives data from Master or Candidate nodes.



# Our Proposal

- Using Raft consensus algorithm If a Master node fails, the Raft election process is initiated to elect a new Master node and avoid single point of failure effects. In this process a node in Candidate state will be elected and will act as Master node, while the previous Master node will switch to Candidate state. Through this process the system maintains its stability and fault tolerance.
- When the Master Controller receives a series of data, a broadcast process is initiated to send such data to all nodes in the network and update information in all controllers accordingly.
- Information is stored in the memory of the Master controller and its initial state is defined as not read.
- Master node sends data to all Candidate nodes and the latter forward such data to their neighboring Worker nodes.
- Master node monitors if all Candidate nodes have received and stores the new data to ensure that all of them have an updated memory.
- Each Candidate node makes sure that it records newly sent data, while it also monitors data records to avoid duplicate ones. Consequently, each Candidate node sends data to attached Worker nodes. During this process, the same approach is followed to ensure successful delivery of data to all nodes.
- When all nodes have successfully forwarded all data, the mechanism transits to an ”OK” state when all controllers have the same data stored in memory.
- When a controller receives new data, then it stops being in the “OK” state and data shall be sent to the other controllers and workers (neighbor nodes) of the mechanism according to the procedure which has been described previously.

# Data input

- Inputs are introduced to the SDN multiple controllers network by clients (nodes). When new data are introduced in the mechanism by a node, according to the Raft protocol, such data is forwarded to the Master Controller. Once the
- Master Controller receives a series of information, it logs and replicates these to all the mechanism controllers, which store such data in their memory.

The main processes that nodes operate upon to maintain consistency, stability, and availability are the following:

- The read process, that reads from the mechanism memory and checks records and if those have been successfully distributed to others.
- The send process that sends data to other controllers.
- The send-to-all process, that is responsible for iteratively sending data to all neighboring controllers.

# Raft Consensus Algorithm in our system

- In the proposed mechanism the main entity is the Raft node which shall be deployed along with each SDN controller in an SDN setup. Each node keeps in its memory a set of records which adhere to the structure record (data, send).
- The variable data holds the information to be exchanged and the variable send is Boolean and is used to flag whether a specific record has been successfully forwarded to the network.
- Nodes are identified by a unique id.
- The Raft consensus algorithm is used to coordinate the sharing of information between nodes. It defines the creation of a group of controllers (candidates) and the required processes to elect one leader the Master controller. The Master is the one who manages the data flow in the mechanism and leads the group if it is active.
- If the Master node fails, then a new Master node must be elected through the mechanism process. Specifically, a time is defined in which the Master sends a message to the other controllers. If the message does not arrive in time, a Controller node sends a message requesting to become the Master node. In this case the other controllers respond, and the specific node is designated as the Master node.

# Performance evaluation
# Experimental Setup

- To evaluate the performance of the proposed mechanism we conducted a network simulation. Also we compare it with other mechanisms in terms of consensus time, distribution time, data access time and presenting test results.
- We have run an experiment to assess the time response of the algorithm. We first run an experimental simulation of a failure scenario in which the proposed algorithm is executed for 100 sec for a mechanism with 30 controllers, 10 of which run as Master nodes and 20 run as Worker nodes. We check that data is being transferred correctly between nodes. In this scenario we assume that at a specific time point, around 20 sec after the start, the master node fails. The main objective is to maintain mechanism stability at all times and avoid the effects of a single point of failure.
- To extend the initial scenario, another test was also executed, in which the newly elected Master nodes drop at time points around 20 sec, 30 sec and 60 sec respectively.
- All nodes have been monitored to test the read and write performance in each node (in Master, Controller or Worker states).
- In the tests, all the nodes except those of the original Master node and the Worker nodes connected to it have the same data after the initially set Master node crashes. Another node is elected as the new Master node. This is repeated a number of times during the execution of the experiment.

# Performance evaluation Results

- Table II shows the basic features of the proposed mechanism in comparison to the CopyCat project and a Paxos-based system that were tested.
  - All models are using consensus algorithms and a distributed SDN multiple controller architecture. As is shown in Table II, through the comparison between the Copycat project and the proposed mechanism.
  - Copycat requires more computing resources than the proposed mechanism and this makes Copycat less reliable for large scale networks.
  - The time required for reading, writing and sending data is higher than the other two mechanisms.
  - The average time that the Copycat system needs to start and elect a Master Controller is 23.23 seconds.

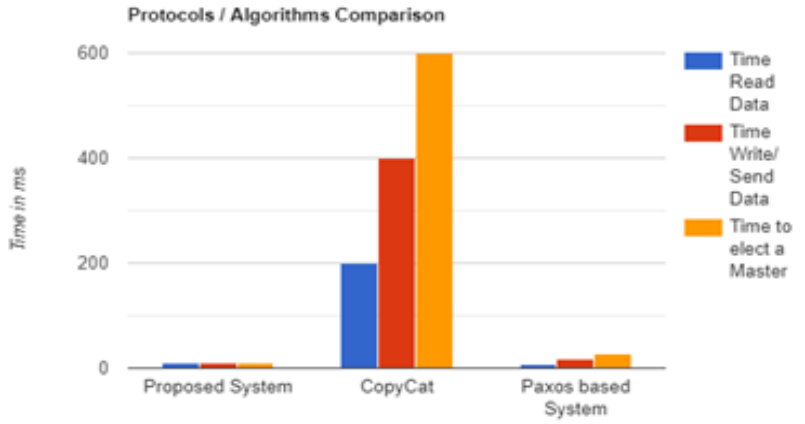
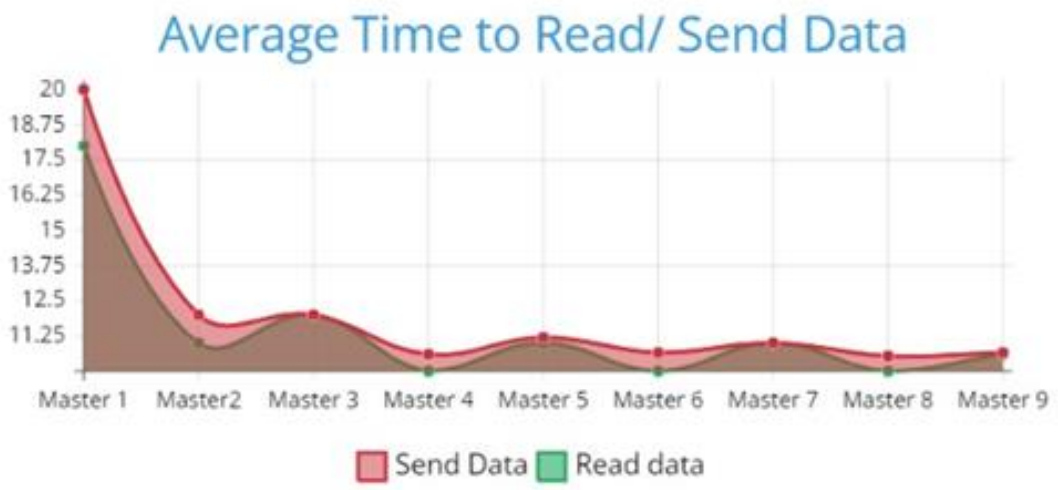



TABLE II
COMPARISON OF PROTOCOLS/ALGORITHMS

| Feature | Proposed | CopyCat Raft | Paxos |
|---|---|---|---|
| Consensus algorithm | Raft | Raft | Paxos |
| Controllers | 10 | 1 | 3 |
| Workers | 2/controller | 1 client | 6 End Users |
| Time to read data | 10 ms | 0.20 s | 6.425 ms |
| Time to write/send data | 10.4 ms | 0.40 s | 17.814 ms |
| Time to elect a Master | 10.06 ms | 0.060 s | 28 ms |

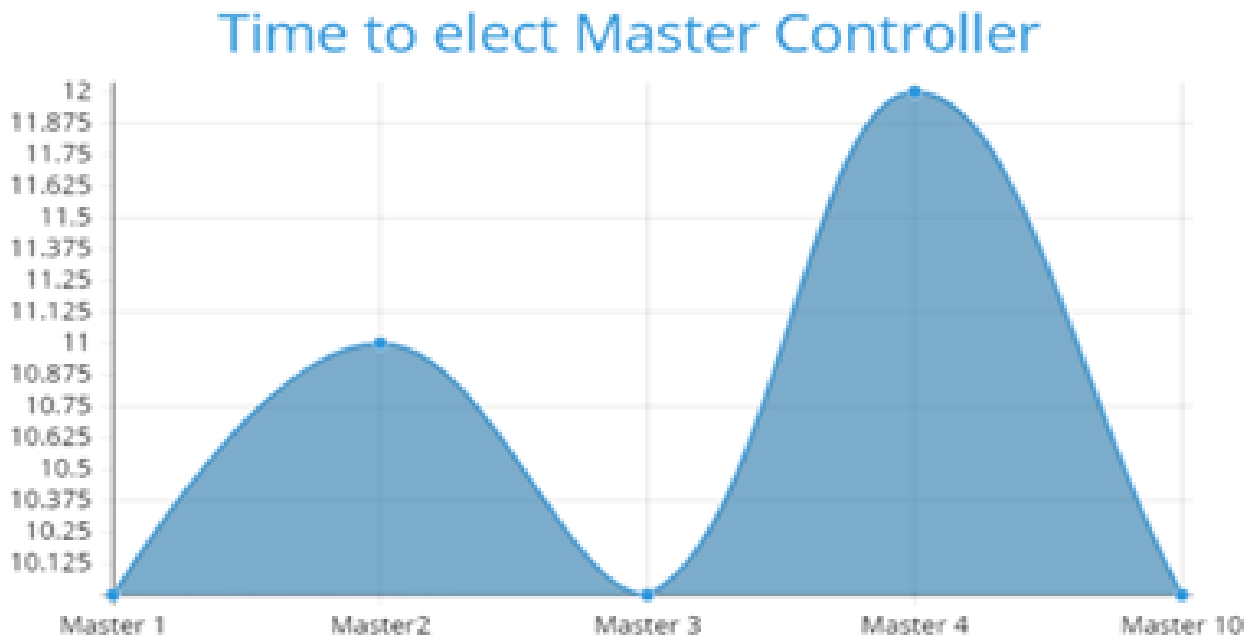
# Performance evaluation Results




- The consensus time of the proposed mechanism is stable.
- Also, the Write/ Send Data time is stable and low the proposed protocol needs 10.4 ms. Low and stable times for reading and sending data can improve the mechanism performance and offer a stable and functional mechanism architecture.
- All controllers had the same data even after a controller drop in our simulation environment of 100 seconds; the network maintains its stability.

# Performance evaluation Results

The test results have shown that for the proposed mechanism the average required time for Master node election is 10.06 ms. It is stable and low, as shown in the figure and described in previous Table.



# Conclusion

- SDN is a promising paradigm for network management because of its centralized network intelligence. However the centralized control architecture of SDNs raises challenges regarding reliability, scalability, fault tolerance and interoperability. The existing solutions which were analyzed in literature are not offering high-throughput, fault-tolerance, and controller synchronization. We proposed a novel implementation, based on the Raft algorithm, that can efficiently synchronize the network state information among multiple nodes, thus ensuring good performance at all times irrespective of the traffic dynamics. Further, the proposed mechanism supports high-throughput, fault-tolerance, and controller synchronization.
- Our simulation results have shown that the proposed mechanism can support Multiple Controllers, as it maintains stability (all nodes have the same data, after a Master node failure) and the average required times are low. The average time it takes to read, write in memory, and send data to neighbor controllers is low and stable. Also, the time it takes to elect a new controller is also low. In our proposal, multiple controllers maintain a consistent global view of the network. This is achieved by employing the Raft consensus protocol to ensure consistency among the replicated network states maintained by each controller.

# References


[1] D. Ongaro and J. Ousterhout, ”In search of an understandable consensus algorithm”, in Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference (USENIX ATC’14), USENIX Association, USA, 2014, pp. 305-–320.

[2] D. Kreutz, et al., ”Software-Defined Networking: A Comprehensive Survey”, in Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[3] L. Lamport, ”The part-time parliament”, ACM Trans. Comput. Syst., vol. 16, no. 2, pp. 133-–169, May 1998, https://doi.org/10.1145/279227.279229.

[4] A. Abdelaziz et al, ”Distributed controller clustering in software defined networks”, PLoS ONE, Vol. 12, no. 4: e0174715, April 2017, https://doi.org/10.1371/journal.pone.0174715(2017).

[5] C. -C. Ho, K. Wang and Y. -H. Hsu, ”A fast consensus algorithm for multiple controllers in software-defined networks,” 2016 18th International Conference on Advanced Communication Technology (ICACT), pp. 1–1, 2016, doi: 10.1109/ICACT.2016.7423293.

[6] D. Dotan and R. Y. Pinter, ”HyperFlow: an integrated visual query and dataflow language for end-user information analysis”, 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC’05), 2005, pp. 27–34, doi: 10.1109/VLHCC.2005.45.

[7] A. Tootoonchian and Y. Ganjali, ”HyperFlow: a distributed control plane for OpenFlow”, in Proceedings of the 2010 internet network management conference on Research on enterprise networking (INM/WREN’10), USENIX Association, USA, 2010.

# References

[8] R. Y. Shtykh and T. Suzuki, ”Distributed Data Stream Processing with Onix,” 2014 IEEE Fourth International Conference on Big Data and Cloud Computing, 2014, pp. 267–268, doi: 10.1109/BDCloud.2014.54.

[9] P. Berde et al, ”ONOS: towards an open, distributed SDN OS”, in Proceedings of the third workshop on Hot topics in software defined networking (HotSDN ’14), Association for Computing Machinery, New York, NY, USA, pp. 1-–6, 2014, https://doi.org/10.1145/2620728.2620744.

[10] S. H. Yeganeh and Y. Ganjali, ”Kandoo: a framework for efficient and scalable offloading of control applications”, in Proceedings of the first workshop on Hot topics in software defined networks (HotSDN ’12), Association for Computing Machinery, New York, NY, USA, 19-–24, 2012, https://doi.org/10.1145/2342441.2342446.

[11] K. Phemius, M. Bouet and J. Leguay, ”DISCO: Distributed multi-domain SDN controllers”, 2014 IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–4, doi: 10.1109/NOMS.2014.6838330.

[12] W. Xia, Y. Wen, C. H. Foh, D. Niyato and H. Xie, ”A Survey on Software-Defined Networking”, in IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, Firstquarter 2015, doi: 10.1109/COMST.2014.2330903.