# Software Defined Network: Load Balancing Algorithm Design and Analysis

Senthil Prabakaran[1] and Ramalakshmi Ramar[2]
[1]Department of Electronics and Communication Engineering, Kalasalingam Academy of Research and Education, India
[2]Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, India

**Abstract:** *Software Defined Network (SDN) cut down the monopolies of producing network devices and their applications. It allows the use of an omniscient controller that manages the overall network and promises for simplifying the configuration and management burden of the traditional Internet Protocol (IP) network. The use of hardware load balancer is a critical issue in conventional IP networks that creates many negative impacts such as the cost affordability, features customization, and availability. Also, the existing load balancing algorithm does not consider the flow size generated by the client nodes. Further, flows are not classified based on the threshold value of the dynamic flow size. The paper proposes to compare the performance of two load balancing algorithms such as flow-based load balancing algorithm and traffic pattern-based load balancing algorithm with distributed controllers' architecture. The result shows that the flow-based load balancing algorithm minimizes response time by 94%, enhances transaction rate by 14% and Traffic pattern-based load balancing algorithm has improved availability by 2.69%.*

**Keywords:** *SDN, distributed controller, flow-based algorithm, traffic pattern based algorithm, failover.*

*Received January 10, 2020; accepted November 24, 2020*
*https://doi.org/10.34028/iajit/18/3/7*

## 1. Introduction

The current traditional Internet Protocol (IP) network is complex and hard to manage Benson *et al*. [4]. The difficulties often arise in the configuration parts, in which each forwarding element must be configured independently by using the manufacturer commands. At the same time, there is a deficiency of an automatic reconfiguration. Furthermore, the vertical integration of the current network is done by the control plane which manages and communicates to the forwarding element. The data plane combined with control plane reduces the flexibility and the innovation Kreutz *et al*. [16]. Controller decouples control plane function from forwarding devices to program and manage the network. By the definition of Software Defined Network (SDN), it refers to the decoupling of the network control part from the forwarding part.

Control, data and application layers characterize SDN architecture. Representational state transfer Application Programming Interface (REST API) manages the control and application layer sessions Alkhatib *et al*. [2]. Openflow switch uses Openflow protocol to send controller instructions to the data plane element Prabakaran and Ramar [23]. Control and data plane communicate the controller instructions using open flow protocol Greene [8]. In SDN nodes are interconnected logically. Thus, using a single interoperable control plane, packet and circuit-switched network can be controlled efficiently Badotra and Panda [3] Figure 1 shows SDN architecture.

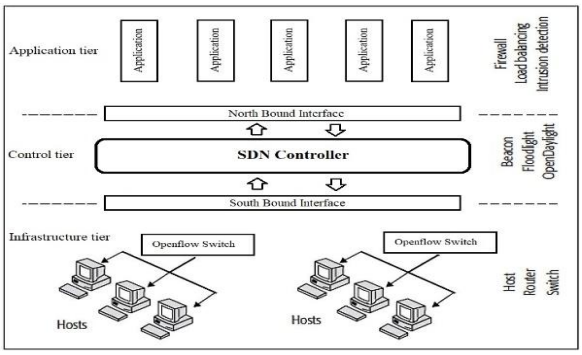


Figure 1. SDN architecture.

In increasing network traffic, load balancing is a critical application to overcome the availability problem and ensure a higher transaction rate with a low response time. Additionally, the load balancing algorithm eliminates and minimize network downtime. Yin *et al*. [29] stated that the controller uses data from flow analysis to enable load balancing decisions. In this paper, the performance comparison of traffic-pattern based and flow-based load balancing algorithms in a distributed controller environment is proposed. POX (An open-source Python based SDN controller application) is the distributed SDN controller [22]. Ahmed and Ramalakshmi [1] proposed SDN distributed controller architecture, which achieves

optimal performance and overcomes the problems like a single-point failure and Network bottleneck, encountered by the centralized SDN controller.

The significant contributions of this paper are:

- Two load balancing algorithms are developed based on traffic flow and traffic pattern on a distributed controller SDN architecture.
- Load balancing algorithm based on traffic flow is implemented to minimize the server's overwhelming problem.
- Load balancing algorithm based on traffic pattern is implemented to reduce processing load and overcome single-point failure.
- The performance of flow-based and traffic pattern-based load balancing algorithms are measured, compared and analyzed using metrics like availability, response time and transaction rate.

The paper is organized as follows: related works in section 2, proposed work in section 3, experimental setup in section 4, simulation results and discussions in section 5, evaluation in section 6 and conclusion in section 7.

## 2. Related Works

A traditional IP network operates with its limitation of implementing a software-based load balancing application entirely due to the API's absenteeism. Thus, conventional network consents for only hardware-based load balancers offer a high performance but with a prohibitive cost and no customization since they are a vendor-locked policy. In this section, various SDN load balancing techniques are examined and classified into:

### 2.1. OpenFlow-Based Load Balancing

Kaur *et al*. [15] proposed a round-robin load balancing algorithm over SDN architecture to distribute incoming traffic evenly to available cluster of servers. The server weight is not considered in this algorithm. Thus, it might be useful in some scenarios in which the capabilities of the servers are equal. Otherwise, some of the servers will be overwhelmed. A path switching method is proposed to overcome the transmission load imbalance problem, which minimizes the response time. The load-balancing algorithm does not consider flow size that could affect the whole process of balancing the load. Similarly, Wang *et al*. [28] implemented the algorithm to match the source IP addresses into the servers. The clients' requests are handled directly by the load balancers. A new transition and partitioning algorithms are proposed to set and change the wildcard rules. It suffers from large overhead that might degrade the controller performance. This architecture considers only the distribution of the incoming traffic without considering the server's side. Thus, the difference in speed and size forces the servers to be overloaded and go out of service. A Load Balancing Based on Server Response Time (LBBSRT) algorithm is proposed by Zhong *et al*. [30] to balance servers load based on response time with centralized controller. LBBSRT selects server with minimum response time. LBBSRT depends only on the real-time measuring without using any techniques to handle the load when the scenario changes. LBBSRT is not a suitable solution for balancing the load when the size of the flows increases.

Least Delay Dynamic Weighted Round-Robin (LDDWRR) by Sroya and Singh [27], implemented a load balancing strategy by assigning the load to the servers based on the delay. LDDWRR shows good response time, throughput, and transactions against the round-robin algorithm. The downside of using LDDWRR is that it will not consider the server's capabilities to handle the weight. Moreover, the weight calculation process is time-consuming, leading to the server's response degradation. Senthil and Ramalakshmi [24] designed Flow-based Proactive Prediction Load Balancing (FPPLB) to quickly and effectively anticipate the current load of the controller. In FPPLB, flows are categorized based on the variations in traffic flows and controller load. Hwang and Tseng [13] proposed an architecture that dealt with switches based on the topology-aware principle and addressed the auto-routing to minimize the human error. On-Line Routing (OLR) calculates the optimal route but it does not address the problem of the flow size.

### 2.2. Dynamic Load Balancing

To manage dynamic traffic burst flows and controller load, a dynamic load balancing algorithm is required. Hamdan *et al*. [10] reported that the controller can make a load calculation and maintain a balance in network load when a predefined threshold exceeds. To improve the overall performance the loads are assigned to specific switches which can handle the flows. Latif *et al*. [17] proposed a dynamic load balancing to reduce latency and optimize the update mechanism based on the information from the data plane element. The FlowBender by Kabbani *et al*. [14] balances distributive flows instead of packets. It used Retransmit Timeout (RTO) to recover from link failure. It relies on Equal Cost Multipath (ECMP) switch support. Likewise, the Mahout by Curtis *et al*. [6] discussed a novel technique which provides efficient visibility of flow behaviour to detect flow size by observing the end hosts' socket. But it suffers from a large overhead that degrades the controller performance. Additionally, Shang *et al*. [26] proposed a load balancing strategy using an adaptive link algorithm and link weight based on the Quality of Service (QoS) principle, which successfully balances the network traffic. The design does not consider the

flow size, which may affect QoS. This technique increases the controller overhead, in which the link weight-based QoS requires to be monitored, and the prioritization needs to occur before sending the traffic. Another limitation is that the algorithm failed to manage the server's load. The link weight-based balancer does not consider server capacity, active connections, and the servers' speed while selecting the backend server. A load balancing by Ma *et al*. [18] decoupled control plane as a meta plane and local plane. The control plane processing is done by local plane and resource management is handled by meta plane.

Hikichi *et al*. [11] used round-robin technique to balance load with distributed controllers to enhance the controller performance. Hai and Kim [9] introduced an algorithm to balance the traffic in a distributed controller network to improve the resource management based on the threshold load on the controllers. This method reduces the communication overhead. Overall controller load is measured, and to improve the efficiency of the network, number of controllers is dynamically extended or narrowed by Nisar *et al*. [20]. Gasmelseed and Ramar [7] developed a load balancing algorithm depending on traffic patterns to divide the packets into transmission control and user datagram protocol, and eventually transmit the flows to the designated controller. Huang *et al*. [12] proposed Congestion Avoidance Video Multicast (CAVM) on SDN environment to monitor bandwidth availability and network link delay to find path with minimum congestion cost and delay.

## 3. Proposed Work

In this section, two load balancing algorithms based on traffic flow and traffic pattern are compared and analyzed in a distributed SDN controller environment.

### 3.1. Load Balancing Application

Load balancer application reduces the network bottleneck problem and increases the network availability. Load balancer application is implemented on distributed and centralized controller environments to compare the performance of the network. If a controller goes down, the secondary controller takes control, to eliminate the network bottleneck problem in distributed environment. Figure 2 shows the load balancer placement on a distributed SDN controlled network.

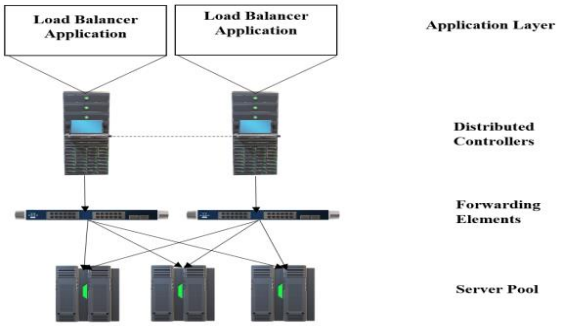


Figure 2. Load balancer application placement.

### 3.2. Load Balancer Algorithm

Load balancers use various algorithms to distribute incoming traffic. Flow-based and traffic pattern-based algorithms are discussed and compared below.

#### 3.2.1. Flow-Based Load Balancer

Figure 3 shows the flow-based load balancer architecture.

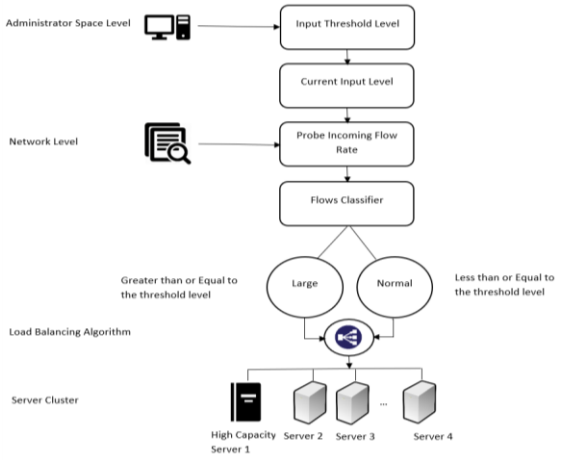


Figure 3. Load balancer architecture based on flow detection.

The flow-based load balancer algorithm deals with the problems of servers overwhelming. The algorithm deals with the absence of real-time monitoring and assigning dynamic threshold level in traditional load balancer applications. The flows are classified into large and normal flows to reduce the processing time taken by the servers and achieve a higher number of transactions. The flow status of controllers is analyzed using sFlow traffic analytic tool [25]. Thus, the loads are distributed evenly between controllers to reduce the overwhelming problem.

#### 3.2.2. Traffic Pattern-Based Load Balancer Algorithm

The load balancing algorithm inspects the ingress packet headers to identify the TCP and UDP packets. The identified packets are forwarded to assigned controllers, and the traffic is distributed to the server pool. Figure 4 shows the load balancer architecture based on traffic pattern.

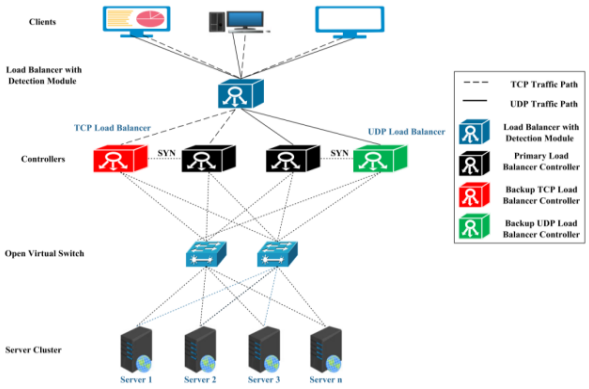


Figure 4. Load balancer architecture based on traffic pattern.

The incoming packet headers are analyzed to find the traffic type. The controller creates and assigns virtual IPs to establish the communication path between the load balancer. It uses source IP to reduce the processing load and minimize the single point failure.

## 4. Experimental Setup

Table 1 shows the experimental setup and parameters

Table 1. SDN setup description and parameters.

| Description | Tools used |
|---|---|
| Network Emulation | Mininet |
| SDN controller type | Distributed |
| SDN controller | POX |
| Forwarding element | OpenVSwitch |
| Flow generator | Curl |
| Servers load generator and tester | Open load and siege |
| Number of servers | 10 |
| Total number of concurrent clients | 500 |

To compare the performance of traffic pattern-based and flow-based load-balancing algorithm with distributed SDN controllers, Mininet is used as network emulator [19]. Open load tool generates load, and siege tool tests the network environment [21]. Curl as a traffic generator to generate traffic from clients to servers [5]. Response time, availability and transaction rate are analyzed and compared to evaluate the proposed work's performance.

## 5. Simulation Results and Discussions

In this section, compared the performance of a traffic pattern-based and a flow-based algorithm considering response time, availability, and transaction rate. Table 2 shows the symbols and descriptions used in the equations.

Table 2. Symbols and descriptions.

| Symbol | Description |
|---|---|
| $S_f$ | Socket failure |
| $\sum c_t$ | Total time taken by the controllers and servers to wait for reply messages |
| Time-out | Time out value of controllers and servers |
| $T_{Trans}$ | The total transaction from clients to servers |
| Tx_time | Overall time taken for transaction |

### 5.1. Availability

The Availability is calculated by considering socket failures and ratio of time out to total time taken by the controller and server as expressed in Equation (1).

$$Availablity = s_f + \frac{Time\_Out}{\sum c_t} \quad (1)$$

In Equation (1), $s_f$ mentions socket failures due to a large number of ingress traffic. The traffic pattern-based algorithm attains higher availability since it has secondary controllers and failover mechanisms, reducing the downtime and bottleneck issue. Figure 5 shows the availability.


Figure 5. Availability.

### 5.2. Response Time

The time taken by the controller to respond to the respective client requests measures the response time. It plays a significant role in a load-balancing environment which has large server and controller clusters. Resulting low response time of an application indicates that the application is well suited for a network environment with a more significant number of clients and request frequency which heavily affect the transaction rate. The flow-based load balancing algorithm achieves low response time compared to the traffic pattern-based algorithm. The reason is the consideration of the flow size, which reduces the standard server from being overwhelmed. Figure 6 shows the average response time.

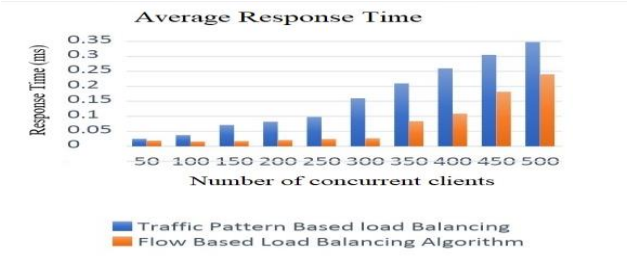


Figure 6. Average response time.

### 5.3. Transaction Rate

The transaction rate is the total number of flows processed by the controllers per second. The transaction rate is inversely proportional to the response time. The transactions rate formula is:

$$Transactions\ Rate = \frac{T_{Trans}}{Tx\_time} \quad (2)$$

Figure 7 shows the transaction rate

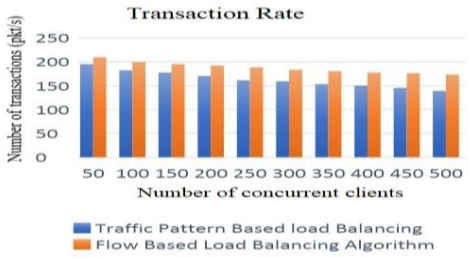


Figure 7. Transactions rate.

The flow-based algorithm achieves a high transaction rate with less response time compared to traffic pattern-based load balancer algorithm.

## 6. Evaluation

As shown in section 5, flow-based load balancing algorithm is best suited for distributed controllers while considering response time and transaction rate parameters. It considers the incoming flows' size and redirects them to the backend servers, consisting of high capabilities and standard servers to process the flows accordingly. In contrast, the load balancing algorithm based on traffic pattern is optimal when addressing the scalability and availability parameters since it uses primary and secondary controllers to ensure availability and growth of the network. Moreover, the load balancing algorithm based on traffic pattern improves the network's performance by using a failover mechanism to eliminate single point of failure problem.

The performance comparison of flow-based load balancing and traffic pattern-based load balancing shows that the flow-based algorithm is more responsive and achieves higher transaction rate. Therefore, it is advantageous for applications like media streaming servers, online gaming servers, Voice over IP (VoIP) servers, Trivial file transfer servers and Virtual Private Network (VPN) tunnelling. The traffic pattern-based load balancing has better network availability. So, it is advantageous for applications like web servers, data farming, remote server and client communication, file transfer servers and mail servers.

## 7. Conclusions

This paper compares the performance of the load balancing algorithms based on traffic flow and traffic pattern. Distributed SDN controller eliminates the disadvantages of a centralized SDN controller architecture to improve the availability, management and scalability of the network. Mininet emulates SDN network, curl, Openload and siege tools are used to generate the traffic load and test the proposed environment. The results display that the load balancing algorithm based on traffic flow has improved the response time by 94% and transactions rate by 14%. Similarly, the load balancing algorithm based on traffic pattern has improved availability by 2.69% due to traffic separation and failover mechanism that lead to achieving better results than flow-based load balancing algorithm.

## References

[1] Ahmed H. and Ramalakshmi R., "Performance Analysis of Centralized and Distributed SDN Controllers for Load Balancing Application," *in Proceedings of 2nd International Conference on Trends in Electronics and Informatics*, Tirunelveli, pp. 758-764, 2018.

[2] Alkhatib H., Faraboschi P., Frachtenberg E., Kasahara H., Lange D., Laplante P., Merchant A., Milojicic D., and Schwan K., "The IEEE CS 2022 Report," *IEEE Computer Society*, pp. 25-27, 2014.

[3] Badotra S. and Panda S., *Handbook of Computer Networks and Cyber Security*, Springer Link, 2020.

[4] Benson T., Akella A., and Maltz D., "Unraveling the Complexity of Network Management," *in Proceedings of 6th USENIX Symposium on Networked Systems Design and Implementation*, Boston, pp. 335-348, 2009.

[5] Curl.Haxx.Se, Command Line Tool, https://curl.se, Last Visited, 2021.

[6] Curtis A., Kim W., and Yalagandula P., "Mahout: Low-Overhead Datacenter Traffic Management Using End-Host-Based Elephant Detection," *in Proceedings of IEEE Infocom*, Shanghai, pp. 1629-1637, 2011.

[7] Gasmelseed H. and Ramar R., "Traffic Pattern-Based Load-Balancing Algorithm in Software-Defined Network Using Distributed Controllers," *International Journal of Communication Systems*, vol. 32, no. 17, pp.e3841, 2019.

[8] Greene K., 10 Breakthrough Technologies: Software-defined Networking, MIT's Technology Review, Last Visited, 2009.

[9] Hai N. and Kim D., "Efficient Load Balancing For Multi-Controller in SDN-Based Mission-Critical Networks," *IEEE in Proceedings of 14th International Conference on Industrial Informatics*, Poitiers, pp. 420-425, 2016.

[10] Hamdan M., Hassan E., Abdelaziz A., Elhigazi A., Mohammed B., Khan S., Vasilakos A., and Marsono M., "A Comprehensive Survey of Load Balancing Techniques in Software-Defined Network," *Journal of Network and Computer Applications*, vol. 174, pp. 102856, 2021.

[11] Hikichi K., Soumiya T., and Yamada A., "Dynamic Application Load Balancing in Distributed SDN Controller," *in Proceedings of 18th Asia-Pacific Network Operations and Management Symposium*, Kanazawa, pp. 1-6, 2016.

[12] Huang H., Wu Z., Ge J., and Wang, L., "Toward Building Video Multicast Tree With Congestion Avoidance Capability in Software-Defined Networks," *The International Arab Journal of Information Technology*, vol. 17, no. 2, pp. 162-169, 2020.

[13] Hwang R. and Tseng H., "Load Balancing and Routing Mechanism Based on Software Defined Network in Data Centers," *International Computer Symposium*, Chiayi, pp. 165-170, 2016.

[14] Kabbani A., Vamanan B., Hasan J., and Duchene F., "Flowbender: Flow-Level Adaptive Routing for Improved Latency and Throughput in Datacenter Networks," *in Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, New York, pp. 149-160, 2014.

[15] Kaur S., Kumar K., Singh J., and Ghumman N., "Round-Robin Based Load Balancing in Software Defined Networking," *in Proceedings of 2nd International Conference on Computing for Sustainable Global Development*, New Delhi, pp. 2136-2139, 2015.

[16] Kreutz D., Ramos F., Verissimo P., Rothenberg C., Azodolmolky S., and Uhlig S., "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, 2014.

[17] Latif Z., Sharif K., Li F., Karim M., Biswas S., and Wang Y., "A Comprehensive Survey of Interface Protocols for Software Defined Networks," *Journal of Network and Computer Applications*, vol. 156, pp.102563, 2020.

[18] Ma Y., Chen J., Tsai Y., Cheng K., and Hung W., "Load-Balancing Multiple Controllers Mechanism for Software-Defined Networking," *Wireless Personal Communications*, vol. 94, no. 4, pp. 3549-3574, 2017.

[19] Mininet.Org, Mininet, http://mininet.org/, Last Visited, 2021.

[20] Nisar K., Jimson E., Hijazi M., Welch L., Hassan R., Aman H., Sodhro A., Pirbhulal S., and Khan S., "A Survey on the Architecture, Application, and Security of Software Defined Networking," *Internet of Things*, vol. 12, pp. 100289, 2020.

[21] Openwebload.Sourceforge.Net, Open Load, http://openwebload.sourceforge.net, Last Visited, 2021.

[22] Pox, The POX Controller, https://github.com/noxrepo/pox, Last Visited, 2012.

[23] Prabakaran S. and Ramar R., "Stateful Firewall-Enabled Software-Defined Network with Distributed Controllers: A Network Performance Study," *International Journal of Communication Systems*, vol. 32, no. 17, pp. e4237, 2019.

[24] Senthil P. and Ramalakshmi R., "Flow Based Proactive Prediction Load Balancing in Stateful Firewall Enabled Software Defined Network with Distributed Controllers," *Journal of Green Engineering*, vol. 10, no. 10, pp. 8337-8355, 2020.

[25] Sflow.Org, Sflow-Flow Monitoring Tool, https://sflow.org/, Last Visited, 2021.

[26] Shang F., Mao L., and Gong W., "Service-Aware Adaptive Link Load Balancing Mechanism For Software-Defined Networking," *Future Generation Computer Systems*, vol. 81, pp. 452-464, 2018.

[27] Sroya M. and Singh V., "LDDWRR: Least Delay Dynamic Weighted Round-Robin Load Balancing in Software Defined Networking," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, pp. 145-148 2017.

[28] Wang R., Butnariu D., and Rexford J., "OpenFlow-Based Server Load Balancing Gone Wild," *in Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, USA, pp. 12, 2011.

[29] Yin P., Diamond S., Lin B., and Boyd S., "Network Optimization for Unified Packet and Circuit Switched Networks," *Optimization and Engineering-Springer*, vol. 21, no. 1, pp. 159-180, 2020.

[30] Zhong H., Fang Y., and Cui J., "LBBSRT: an Efficient SDN Load Balancing Scheme Based on Server Response Time," *Future Generation Computer Systems*, vol. 68, pp. 183-190, 2017.

**Senthil Prabakaran** was born at Dindigul, India, in 1987. He graduated in Electronics and Communication Engineering from Anna University affiliated college and post graduated in Network Engineering from Kalasalingam Academy of Research and Education, Krishnankoil, India. He is pursuing his PhD in Electronics and Communication Engineering (Software Defined Networking) from Kalasalingam Academy of Research and Education. His research interest includes Computer Networks, Software Defined Networks, Cloud Computing, Network Function Virtualization and Network Security.

**Ramalakshmi Ramar** received her Doctoral degree and Master of Engineering degree in Computer Science and Engineering. She has been working in the department of Computer Science and Engineering at Kalasalingam Academy of Research and Education (Previously known as Arulmigu Kalasalingam College of Engineering) since 2001. She has more than 20 years of teaching experience. She is a member of CSI, ISTE and Network Technology group of TIFAC- CORE in Network Engineering. She has published more than 25 research articles in reputed Journals and International Conferences. She has received Young Scientist Fellowship from Tamilnadu State Council for Science and Technology and Award of Excellence from SAP India Pvt. Limited. Her areas of research include Software Defined Networking, Cognitive Science, Internet of Things, Big Data Analytics and Social Network Analysis.