A Hybrid Approach for Providing Improved Link Connectivity in SDN

Muthumanikandan Vanamoorthy[1], Valliyammai Chinnaiah[1], and Harish Sekar[2]
[1]Department of Computer Technology, Madras Institute of Technology, Anna University, India
[2]Endurance International Group, India

Abstract: *Software-Defined Networking (SDN) is a unique approach to design and build networks. The networks services can be better handled by administrators with the abstraction that SDN provides. The problem of re-routing the packets with minimum overhead in case of link failure is handled in this work. Protection and restoration schemes have been used in the past to handle such issues by giving more priority to minimal response time or controller overhead based on the use case. A hybrid scheme has been proposed with per-link Bidirectional forwarding mechanism to handle the failover. The proposed method makes sure that the controller overhead does not impact the flow of packets, thereby decreasing the overall response time, even with guaranteed network resiliency. The computation of the next shortest backup path also guarantees that the subsequent routing of packets always chooses the shortest path available. The proposed method is compared with the traditional approaches and proven by results to perform better with minimal response time.*

Keywords: *Open flow, SDN, link failure, protection and restoration.*

Received October 12, 2018; accepted January 21, 2019
https://doi.org/10.34028/iajit/17/2/13

1. Introduction

Software-Defined Networking (SDN) is revoultionalizing the way to architecture networks. It provides numerous advantages in terms of automated load segregation, traffic monitoring and virtualization. It has 2 major planes-control plane and data plane. Control plane is the centralized unit which is programmed to direct the routing process in the data plane. SDN allows easy reconfiguration since the intelligence of the network is centralized. The end to end connectivity among the nodes is monitored by the SDN controller. It is also capable of computing and monitoring the overall throughput and response time.

Open flow is an open source implementation of the protocol and acts as an interface between switches and controllers. The centralized control over the network is handled by open flow by remotely controlling the forwarding table of all the switches. There are three main components in the openflow architecture namely data plane, control plane and a secure communication channel to link them.

Each open flow switch has three tables to manage the process of forwarding the packets. These are flow table, group table and meter table. Flow table has a list of entries based on which it reacts to the incoming packet and takes the action to forward it accordingly. There can be multiple entries/rules to control the forwarding process and these rules will be sorted on based of the priority of the rules. The secure channel will be used to pass packets to the controller when none of the forwarding rules match with the packet. Some algorithms may even choose to drop such packets.

Headers, counters and action are the three important fields of a flow entry. The duration of active links, count of received packets and count of transmitted packets are among the frequently encapsulated fields in counters. The action field chooses the forwarding path of the matched packet. To handle multiple flows, group table can be used.

Group table is organized in buckets which have more than one rule per incoming packet. Meter table is used to control the performance of a switch. Virtualization of a network can optimize the computation and storage capacity of the network. These features play an important role in automating the network functionalities, thereby reducing functional costs.

Fault tolerance of a network can be defined as the capability to swiftly respond and handle an unexpected failure. Network protection to handle link failures and reroute the packets to the destination is significant. This requires the network to maintain the active links so that the backup paths are computed accordingly to forward the packets.

SDN provides the ability to control the bandwidth based on the requirements. The underlying network is managed by the controller to enhance the network performance. SDN applications can be operated in multiple variations like Software Defined Mobile Networking (SDMN), Software Defined Local Area Networking (SD-LAN) and Software Defined Wide Area Networking (SD-WAN). Server virtualization in large data centres provided by SDN is seen as a

primary advantage to handle large networks. These features of SDN make sure that it reduces operational costs and enhances network visibility.

The paper is partitioned into seven sections. The second section covers the literature survey of the mechanisms used so far in the past works, the third section explains the proposed work, the fourth section details on the mathematical justification, the fifth section briefing the experimental setup followed by the sixth section which shows the results and the final section covers the conclusion and future work.

2. Literature Survey

Optimization of the flow restoration process has been handled in the past [1]. Two cases of handling the flow operation were studied-1) Add and Remove, 2) add-only. The proposed algorithm was targeted to find the near to optimal solution. But, the technique did not focus on minimizing the response time.

Bidirectional Forwarding Detection (BFD) had been closely associated to handle node and link failures effectively in the past [12, 18]. It involves a three-way handshake process for establishing the session. It was concluded that BFD can be used as an efficient Operation, Administration and Maintenance (OAM) mechanism for IP RAN networks to serve the needs of node and link fault tolerance with minimum overhead.

Fast failover had been handled using Proactive mechanisms [8, 10, 15] in software defined networks. Controller was frequently involved in the failover process using the protection scheme. Flow aggregation had been used as its main principle using the Virtual Local Area Network (VLAN) tag id in order to provide fast failover. The recovery time was low in this process, thereby enabling this mechanism to be feasible for carrier grade networks.

The issue of robust multicasting had been treated with the open flow framework [2]. Reactive and Proactive are two major ways of handling fault tolerance. To provide proactive fault tolerance, a multicast tree was involved which connected one publisher to a set of subscribers to provide resilience. The packet information was carried using the VLAN tags. The scope of the work did not involve the reactive mechanism to handle failover and switchover.

Load balancing and congestion control are the two other important aspects of network performance in data center networks [9, 11]. Software defined networking provides an additional advantage of handling dynamic load balancing and multipath forwarding to solve the problem of congestion control. Effective traffic control had been done using the notification mechanism among the switches. It was an economical approach to handle the increasing network loads in data center networks.

SDN switches and traditional IP routers can co-exist in a hybrid mechanism which can be used to solve the issue of single link failure and guarantee traffic reachability [5]. IP tunnelling protocols were used to implement this approach. Computed results had shown that the number of SDN switches required for this approach were comparatively lesser. The approach was better in terms of performance when compared to the shortest path recalculation approach.

Failover mechanism was introduced with per-link BFD [16]. Group tables and BFD were combined to provide superior link failure detection. The recovery time of this approach was independent of the size of network and length of path.

A framework for Open State had been used to handle both node and link failures [3]. It made use of the protection scheme using pre computed alternate paths. Mixed Integer Linear Programming (MILP) process was formulated to optimize fast-failover reroutes and path calculation for all potential link failures. The ways in which link failure would arise were not considered in this approach.

Multiple link failures frequently seen in carrier networks have also been solved in the past [14]. The authors proved that for any small number of edge *k*, the network design, related protocol, and backup path reconfiguration scheme could handle *k* arbitrary link failures and provided no loss of connectivity and congestion. The existing link-based restoration using FRR is fast but forms congestion which leads to packet loss for sensitive applications by overloading edges.

A novel network protection mechanism, called Independent Transient Plane (ITP) was proposed where there were two uncorrelated planes -working and transient planes [7]. The transient plane was used to route the packets to overcome the disadvantages of segment protection [13]. The design managed to reduce the flow table entries by 60% and used minimal number of configuration messages. But, the mechanism was designed only for OpenFlow based networks. Mechanisms designed for providing fault tolerance were also focussed on particular types of networks such as grid [6] to improve the application turnaround time.

Congestion-Aware Local Fast Reroute (CALFR) had been designed by leveraging flexible flow aggregation in fast reroute to balance failure recovery time and forwarding rule occupation [4]. The problem was formulated as an integer linear programming model. In node failure of SDN, Bellman Ford and Dijikstra algorithms were used for providing resolution for loop confrontation and bandwidth allocation of nodes respectively [17].

The literature survey can be summarized as follows:

1. Restoration and Protection schemes are two frequently used failover mechanisms.
2. Protection method is focussed on adding the entries in the flow table beforehand, thereby avoiding the

controller overhead in case of a link failure.
3. Restoration method involves the controller call whenever there is a link failure. This ensures smaller flow table but there is a overhead involved in the controller call affecting performance.
4. BFD is an effective algorithm for detecting the status of links.
5. Numbers of configuration messages, size of flow table and response time are the most important factors in determining the performance of a network.

3. Proposed Work

3.1. Architecture

The proposed system architecture is drawn in Figure 1. The SDN controller which acts as the centralized base of intelligence encapsulates three components. It collaborates between the three units and accomplishes the rerouting process. The three components are namely real time controller, link failure handler and link failure detector.

The SDN controller communicates using the OpenFlow protocol with the switches. Each open flow switch is associated with a flow table and a group table. There is a bidirectional interaction described in the figure to demonstrate the interaction between the switches and controller. This is due to the fact that the switch informs about the failed packet delivery using the faulty link to controller and the controller installs the new flow entries after the computation of the new shortest paths.

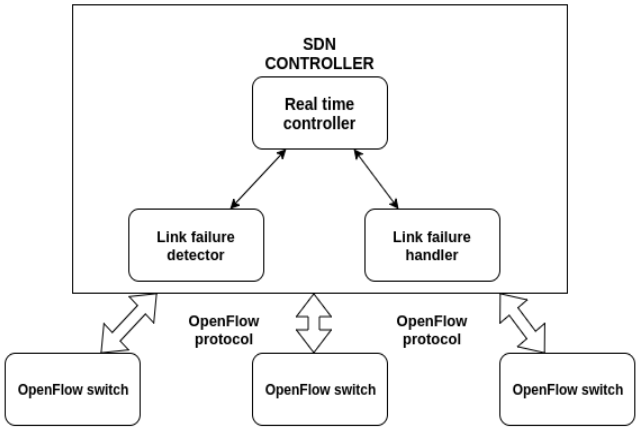

Figure 1. Proposed hybrid system architecture.

The real time controller is the centralized component of the SDN controller and it functions to collaborate between the link failure detector and handler. When a faulty link is notified by a switch via the secure control channel, the real time controller establishes asynchronous BFD switches to detect the status of other links. The source switch sends a control message and waits for the acknowledgement in a particular timeframe. Once the status of the links are figured out by the link failure detector, it transfers the control to the real time controller which proceeds forward to handle it using the link failure handler.

The link failure handler proceeds to compute the shortest path in the current topology using the configured algorithm and sends the computed details to the real time controller. The real time controller sends messages to the switches to install the new flow and group entries. The group entries make sure that the protection part of the process is covered, to make sure that the packets can again be forwarded without controller intervention when there is a similar link failure in the future.

3.2. Network Protection Against Link Failure

Link failure detection is the first step of the hybrid approach. It is significant to detect the status of all links from the source switch that reported the failure. It is because of the fact that subsequent routing should be optimized for the topology at that point of time. It uses BFD for this process. It is operated in two modes-Asynchronous or demand. Here, asynchronous model is used and the source switch sends an asynchronous control message. Timers are set and the source switch waits for the response from all of its immediate neighbours to detect the link availability. Based on the results obtained from this process, the current topology is determined.

Fault handling is handled in a hybrid manner including both the restoration and protection schemes. The process involves two steps. The switch uses the pre installed backup entries to determine the path for the delivery of the current packets. However, to make sure that the future path remains optimized, it makes the controller call and the controller computes shortest path using Dijikstra's algorithm in the second step and transmits the flow entries to the corresponding switch. It can be used in case of future link failures and thus it is the entry that we make use of in the first step. If there is no backup entry configured in the group table of the switch, it traces back the packet to the source to check if there is any path from it and this process keeps repeating recursively.

Consider the topology in Figure 2. There are a couple of hosts and six switches. Assume an use case where the packet transmission is from host 'src' to host 'des'. The flow and group entries are pre configured in the switches along with the backup entries to handle the failure in the primary path. In the normal scenario, the packets will be transmitted in the primary shortest path 1-2-3. However, keeping link failures in mind to provide fast failover, there are group entries in all the switches to its corresponding destination node. For example, switch 6 has two group entries pertaining to port 2 and 3 which acts as primary and backup entries for the potential failure in link 6-3. Therefore the switch will transmit the packets in the 1-6-4-3 path if the link 6-3 fails.

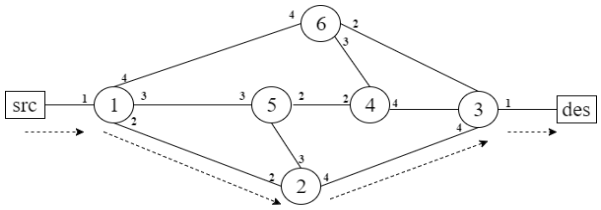

Figure 2. The Topology of a sample network.

The failure of the link 2-3 is shown in Figure 3. Using the group table entries, the packets start flowing in 1-2-5-4-3.

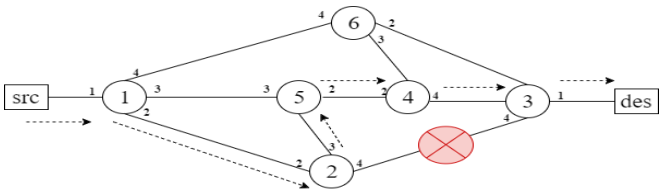

Figure 3. Fast failover process using group table.

Even though fast failover is provided using the group entry, the path used for the transmission might not be the most efficient path for future packets to flow. Here, the path 1-6-3 is up and hence the shortest, once the link 2-3 goes down. Thus the controller takes the responsibility to figure out the least distance path once it gets the notification from the switch 2. The path 1-6-3 is found to be optimal and the controller proceeds to install the new entries as shown in Figure 4.

The controller also adds the backup entries for each failed link in the new primary path. Therefore to account for failure in link 1-6, the transmission 1-5-4-3 is identified and similarly to account the failure of link 6-3, the transmission 6-4-3 is identified. Hereby, we guarantee the most optimal path in the future transmissions while providing instant fast failover.

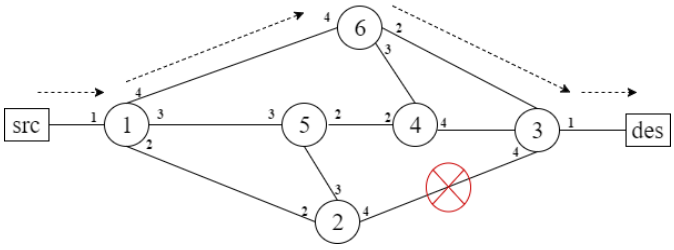

Figure 4. Computation of the current shortest path.

The immediate fast failover approach and long term optimal routing are both handled appropriately. The controller action is described in the following algorithm.

Algorithm 1:controller_action(packet, switch, link_failed)

```
Input: Source switch with the failed link
Output: Optimal path installed in the switches
Declare and initialize path [][]={0}
for each interface in switch
{
 Enable multiple BFD async sessions recursively
    Initialize timers and proceed on the sessions
    Pass the control message
    Trace the response within TTL and store in response
    if response
      {
       Insert the link in path[][]
      }
}
shortest_path = dijikstra(path, src, dest)
insert_flow_entries(shortest_path)
for each link in shortest_path
{
    path = path – {link}
    current_path = dijikstra(path,link.src,dest)
    insert_flow_entries(current_path)
    path=path+{link}
}
```

The switch functions as per the following algorithm.

Algorithm 2: switch_action(packet)

```
Input: Transmission packet
Output: Invoke controller or transmission of packet
output_port = get_next_hop(flow_table, input_port);
if output_port is up
{
    transmit the packet in output_port
    return
else
    backup_port= get_backup(flow_table, input_port)
    transmit the packet in backup port
    controller_action(packet,this.name,output_port)
}
```

4. Analysis of Flow Entries

The analysis of flow entries is performed with the traditional open flow ring topology. Let us assume that the count of hosts interfaced to each switch is denoted by *M* and the number of switches is denoted by *N*. The average count of flow entries can be evaluated using the expression Equation (1).

$$W = N*(N-1)*M + M*N \quad (1)$$

The Equation (1) can be equated down into Equation (2).

$$W = (N\hat{}2) *M \quad (2)$$

The above expression, as each switch needs to have entries that matches packets to the hosts connected to itself, the second term of the expression *M*N* is the number of such flows. In order to match other hosts connected to other switches, each switch must match the IP address of the destination using a flow entry. As (*N*-1)**M* denotes the count of number of hosts apart from those connected to that switch, the total number of such entries is denoted by the expression *N**(*N*-1)**M*.

The flow entry's action can be based on the IP address of the input switch or the port. The above equation denotes the flow entries when Internet Protocol address is considered. But, usually the port number is efficient in terms of lesser flow entries as it can aggregate many flow entries into a single bucket in the flow table. The average count of working flow

entries based on the action using port number is denoted in the expression as Equation (3).

$$W = 2*N + M*N \quad (3)$$

As there are only 2 flows required for aggregation based on the port number in the ring topology, 2*N is added with the other entries needed to send packets targeted at the hosts connected directly to itself as seen in the first case. It must be noted that this equality may not be the same with respect to any other topology.

5. Experimental Setup

Mininet network simulator and Opendaylight SDN controller have been used to simulate the algorithm. The hybrid failover algorithm is implemented using Python invoking the Opendaylight libraries and Application Programming Interface (API's). The network topology is designed in Mininet in Python.

The Operator Discretization Library (ODL) libraries primarily used were dlux and l2switch. The Virtual Machine's (VM) operating system used to implement the system is Ubuntu 16.06 on top of the Windows 10 host OS using Oracle Virtual Box. The Random Access Memory (RAM) allocated for the VM is 2 GB. The Opendaylight controller operates in the port number 6633.

6. Implementation and Results

The response time of the proposed hybrid fast failover method is compared with the traditional restoration scheme. The comparison of the response time is depicted in Figure 5. As depicted, since the controller is involved in the restoration approach, the response time grows with the number of flow entries, but in the proposed hybrid approach since the failover depends only on the backup path, there is not a significant rise in the response time.

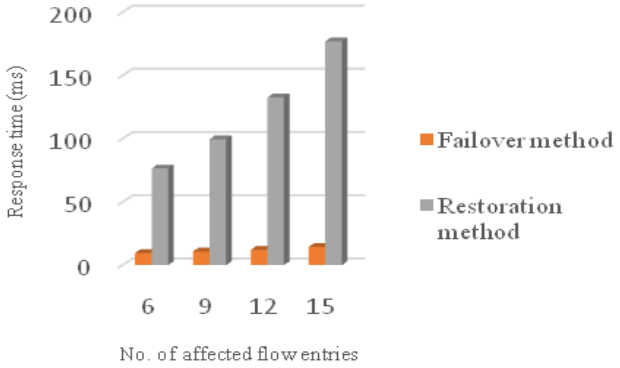

Figure 5. Response time.

The number of backup flow entries is plotted against the number of switches in Figure 6. The segment based protection method is compared with the proposed hybrid approach. The experimental result suggests that restoration method needs more entries than the proposed method. Backup entries play an important role while routing the packets in case of a failure in the primary path. Therefore, number of flow entries is an important performance attribute. Clearly, the proposed method overcomes the segment based approach.

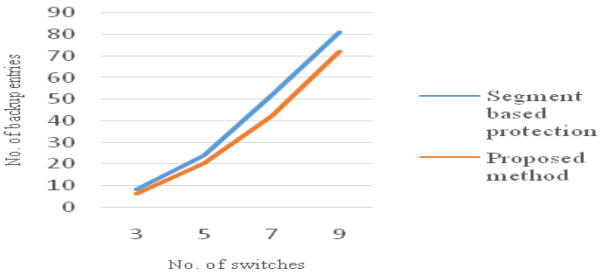

Figure 6. Backup flows entries.

The response time required to forward the packet is plotted against the number of switches in Figure 7. Failover and switchover response time play an important role in judging the performance of an algorithm. Failover is the time required to forward the packet to the destination while switchover is the computation time taken by the controller to compute the subsequent future path. Thus the graph compares the failover and switchover time with the increase in number of switches.

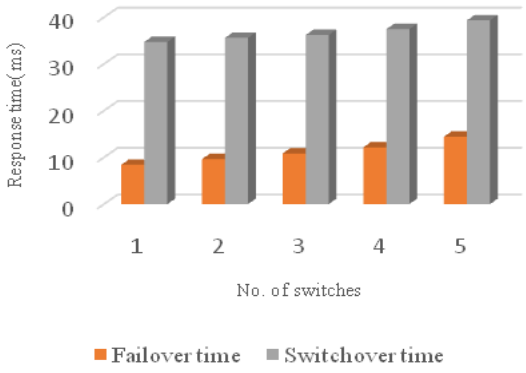

Figure 7. Failover and switchover time comparison.

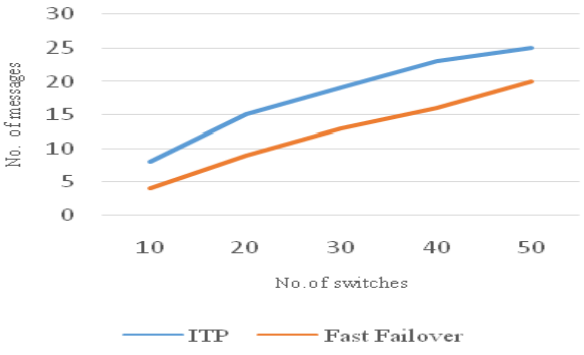

Figure 8. Number of configuration messages.

The most recent design for providing link fault tolerance is based on independent transient plane. The ITP method is compared with the proposed approach based on the number of configuration messages

against the number of switches in Figure 8. Configuration messages are the control messages transmitted using the secure control channel. These messages play a crucial role in the coordination between the switches and controller. This includes the faulty link notification messages and the flow entry installation messages.

7. Conclusions and Future Work

A hybrid approach for the delay-less transmission of on-the-fly packets to the destination with minimal controller overhead and response time has been discussed and this method also makes sure that the future flow of packets would always take the shorter path if available in the updated topology. The key features of the method can be summarized as follows.

1. Fast failover is guaranteed in the proposed approach
2. Optimal path in the longer run is assured with on-the-fly packets delivered using the available backup path
3. BFD is efficiently used to compute the status of the links and arrive at the current topology of the system
4. The proposed approach is superior to the traditional restoration and protection schemes since it provides the advantages of both of these mechanisms.

The classification of link failure types into permanent and transient failures can play an important factor in determining the fast failover approach. SDN protocol implementations other than open flow can also be studied and experimented with the proposed approach. Congestion factor is one factor that can be used to extend the proposed approach in the re-routing process, where the new path will depend on the traffic load that the links are facing. It can be used to prioritize the backup link when there are multiple options with the same cost. Packets which are identified to be malicious can be handled differently during the routing process. These packets can either be dropped or passed to the controller via the secure channel for the controller to make a choice on whether to forward the packet via the primary path or not.

References

[1] Astaneh S. and Heydari S., "Optimization of SDN Flow Operations in Multi-Failure Restoration Scenarios," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 421-432, 2016.

[2] Ahmed R., Alfaki E., and Nawari M., "Fast Failure Detection and Recovery Mechanism for Dynamic Networks Using Software-Defined Networking," *in Proceedings of IEEE Conference of Basic Sciences and Engineering Studies*, Khartoum, pp. 167-170, 2016.

[3] Capone A., Cascone C., Nguyen A., and Sanso B., "Detour Planning for Fast and Reliable Failure Recovery in SDN with OpenState," *in Proceedings of IEEE International Conference on the Design of Reliable Communication Networks*, At Kansas, pp. 25-32, 2015.

[4] Cheng Z., Zhang X., Li Y., Yu S., Lin R., and He L., "Congestion-Aware Local Reroute for Fast Failure Recovery in Software-Defined Networks," *Journal of Optical Communications and Networking*, vol. 9, no. 11, pp. 934-944, 2017.

[5] Chu C., Xi K., Luo M., and Chao H., "Congestionaware Single Link Failure Recovery in Hybrid SDN Networks," *in Proceedings of IEEE Conference on Computer Communications*, Kowloon, pp. 1086-1094, 2015.

[6] Khan M., Hyder S., Ahmed G., and Begum S., "A Group Based Fault Tolerant Scheduling Mechanism to Improve The Application Turnaround Time on Desktop Grids," *The International Arab Journal of Information Technology*, vol. 13 no. 2, pp. 274-280, 2016.

[7] Kitsuwan N., McGettrick S., Slyne F., Payne D., and Ruffini M., "Independent Transient Plane Design for Protection in OpenFlow-Based Networks," *Optical Communication Networks*, vol. 7, no. 4, pp. 264-275, 2015.

[8] Kitsuwan N., Payne D., and Ruffini M., "A Novel Protection Design for OpenFlow-Based Networks," *in Proceedings of 16th International Conference on Transparent Optical Networks*, Graz, pp. 1086-1094, 2015.

[9] Li J., Hyun J., Yoo J., Baik S., and Hong J., "Scalable Failover Method for Data Center Networks Using Open Flow," *in Proceedings of IEEE International Conference*, Krakow, pp. 1-6, 2014.

[10] Lin Y., Teng H., Hsu C., Liao C., and Lai Y., "Fast Failover and Switchover for Link Failures and Congestion in Software Defined Networks," *in Proceedings of IEEE Communication QoS, Reliability and Modeling Symposium*, Kuala Lumpur, pp. 1-6, 2016.

[11] Mallik A. and Hedge S, "A Novel Proposal to Effectively Combine Multipath Data Forwarding for Data Center Networks with Congestion Control and Load Balancing Using Software-Defined Networking Approach," *in Proceedings of IEEE International Conference on Recent Trends in Information Technology*, Chennai, pp. 1-7, 2014.

[12] Nordell V., Gavler A., and Skoldstrom P., "BFD Triggered, GMPLS Based Multi-Layer Ethernet Access Network Protection," *in Proceedings of IEEE International Conference Asia*

Communication and Photonics Conference and Exibition, Shanghai, pp. 1-6, 2011.

[13] Sgambelluri A., Giorgetti A., Cugini F., Paolucci F., and Castoldi P., "Open Flow-Based Segment Protection in Ethernet Networks," *Journal of Optical Communications and Networking*, vol. 4, no. 9, pp. 1066-1075, 2013.

[14] Sinha R., Ergun F., Oikonomou K., and Ramakrishnan K., "Network Design for Tolerating Multiple Link Failures Using Fast Re-Route," *in Proceedings of IEEE International Conference on the Design of Reliable Communication Networks*, Ghent, pp. 1-8, 2014.

[15] Thorat P., Challa R., Raza S., Kim D., and Choo H, "Proactive Failure Recovery Scheme for Data Traffic in Software Defined Networks," *in Proceedings of IEEE Netsoft Conference and Workshops*, Seoul, pp. 219-225, 2016.

[16] Van Adrichem N., Van Asten B., and Kuipers F., "Fast Recovery in Software-Defined Networks," *in Proceedings of 3th European Workshop on Software Defined Networks*, London, pp. 61-66, 2014.

[17] Waleed S., Faizan M., Iqbal M., and Anis M., "Demonstration of Single Link Failure Recovery using Bellman Ford and Dijikstra Algorithm in SDN," *in Proceedings of International Conference on Innovations in Electrical Engineering and Computational Technologies*, Karachi, pp. 1-4, 2017.

[18] Zheng R. and Sun S., "BFD-triggered OAM Mechanisms for IP RAN Network," *in Proceedings of IEEE Zheng International Conference on Electronics Information and Emergency Communication*, Beijing, pp. 286-288, 2013.

Muthumanikandan Vanmoorthy is currently working as a Teaching Fellow at the Department of Computer Technology, Madras Institute of Technology Campus, Anna University, Chennai, India. He received his B.E and M.E degree in Computer Science and Engineering discipline. He received his PhD in CSE from Anna University. His areas of interests include Networking, Software Defined Networking and Network Function Virtualization.

Valliyammai Chinnaiah is an Associate Professor in the Department of Computer Technology, Madras Institute of Technology Campus, Anna University, Chennai, India. She received her Ph.D. in computer science and engineering at Anna University. She has 15 years of teaching experience. Her areas of interest include Cloud computing, Big Data, Network management, Grid computing and Mobile agents. She has published around 46 papers in National and International conferences and journals.

Harish Sekar is a Computer Science Engineer and he obtained his Bachelor's degree from the Madras Institute of Technology, Chennai from the Department of Computer Science and Engineering. He is currently associated with Endurance International Group as a Software Engineer at Bengaluru, India. Harish is an expert coder and has a lot of passion for algorithms. He has done many projects and has strong interest in SDN.