Adaptive Recovery Mechanism for SDN Controllers in Edge-Cloud supported FinTech Applications

Xiaodong Ren*, Gagangeet Singh Aujla†, *Senior Member, IEEE,* Anish Jindal‡, *Member, IEEE,* Ranbir Singh Batth‖, *Member, IEEE,* Peiying Zhang¶

***Abstract*—Financial Technology have revolutionized the delivery and usage of the autonomous operations and processes to improve the financial services. However, the massive amount of data (often called as big data) generated seamlessly across different geographic locations can end end up as a bottleneck for the underlying network infrastructure. To mitigate this challenge, software-defined network (SDN) has been leveraged in the proposed approach to provide scalability and resilience in multi-controller environment. However, in case if one of these controllers fail or cannot work as per desired requirements, then either the network load of that controller has to be migrated to another suitable controller or it has to be divided or balanced among other available controllers. For this purpose, the proposed approach provides an adaptive recovery mechanism in a multi-controller SDN setup using support vector machine-based classification approach. The proposed work defines a recovery pool based on the three vital parameters, reliability, energy, and latency. A utility matrix is then computed based on these parameters, on the basis of which the recovery controllers are selected. The results obtained prove that it is able to perform well in terms of considered evaluation parameters.**

***Index Terms*—Classification, Controller recovery, Financial Technology, Software-defined networks, Support vector machine.**

I. Introduction

Financial Technology (FinTech) has evolved as a way to automate and improvise the delivery and use of financial services thereby enabling the business organisation and consumers to manage their financial operations and processes effectively. Initially, FinTech was limited to the back-end services but with the rise of consumer-oriented applications, it is now seen as a major revolution for the consumers related to financial applications (such as retail banking, investment management, etc). Thus, FinTech can be seen as a synergy between finance and technology that can act as a software, a service, or a business that utilises technological advancements like 1) Artificial Intelligence (AI) and Machine Learning (ML), 2) Big Data and Data analytics, 3) Robotic Process Automation (RPA), and 4) Blockchain [1], [2]. The major application and enablers for FinTech are shown in Fig. 1. Some of the major applications of FinTech are discussed below [2].

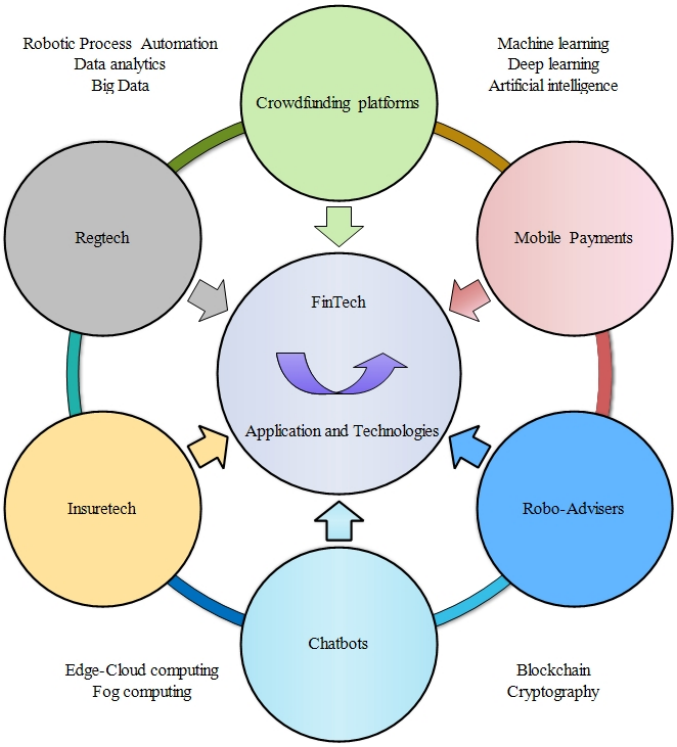

Fig. 1. FinTech Applications

- *Crowdfunding platforms:* These platforms (like Kickstarter, GoFundMe, etc) enables early-stage businesses and entrepreneurs to raise funds or capital form international market, bypassing all geographic limitations.
- *Mobile Payments:* The mobile payment gateways are another application of FinTech that are most prevalent in the current times (even helped in COVID pandemic situation). These gateways enable the consumers to carry the routine banking activities using their mobile phones without even visiting the bank premises physically.
- *Robo-Advisers or Chatbots:* These online investment managers or virtual assistants that are based on AI or ML algorithms to help the customers with their queries and allocate assets accordingly. The Robo-advisers are popular among the investment sector for providing investment service and activities at a minimal cost.
- *Insuretech:* This platform relates to the new insurance model that provides a secure and tailored services thereby

X. Ren is with the School of Automation Science and Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, P.R. China. E-mail: rxd@xjtu.edu.cn

G.S. Aujla is with the School of Computing Science, Durham University, United Kingdom. E-mail: gagi_aujla82@yahoo.com

A. Jindal is with the University of Essex, Colchester, United Kingdom. E-mail: a.jindal@essex.ac.uk

R.S. Batth is with the School of Computer Science Engineering, Lovely Professional University, India. E-mail: ranbir.21123@lpu.co.in

P. Zhang is with the College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, P.R. China. E-mail: 25640521@qq.com

streamlining the insurance process, filling claims and managing the policies online.
- *Regtech:* This platform helps to automate the compliance process for relulating the financial organisations in a quick and cost-effective manner.

These applications imply that FinTech has converted the entire financial sector into a data-driven platform where a massive data is generated continuously, starting from the initiation of a financial service, transmitting the data using network backbone, processing it using computing resources (mostly cloud-based data centers), extracting the knowledge using learning algorithms, passing the desired information to the customer, and storing the relevant data on backup cloud data stores. To handle these applications, the Cloud offers variety of services in terms of shared and abundant network resources to the users such as-storage, computing, and infrastructure [3]. The major advantage of cloud is its ability to provide quality of service (QoS) to the end-users [4].

However, as the FinTech applications are distributed with a range of small to medium to large service scenarios so processing the data at the Cloud may not serve the latency requirements of financial services like mobile payments. But, the cloud has evolved over the time to edge and fog computing which can provide financial services closer to the end-users premises [5], [6]. These computing paradigms compliment the cloud environment to form an edge-cloud ecosystem which is fast, efficient, and provides improved quality of experience (QoE) [7]. The dependence of a larger consumer base through connected devices (such as-phones, computers, PDAs, etc.) generate humongous data which needs real-time transmission and processing using edge-cloud infrastructure. This puts the underlying network under tremendous load. So, to maintain the efficiency in such a financial network, a scalable and reliable network architecture is required.

A. Network Attributes for FinTech

In a FinTech environment, the desired network characteristics are low latency, high bandwidth, high bit rate, and negligible congestion [8], [9], [10]. The major network attributes desired to support FinTech clusters are listed below [11], [12], [13], [14], [15].

- *Predictable and Efficient*: Traffic patterns and characteristics are likely to be variable and bursty due to the uncertain nature of the data traveling across the network at any time and scale. An adaptable and predictive network can enhance the overall efficiency and reduce the latency in large data clusters.
- *Holistic Network*: A network which can manage the transactions between massive parallel servers with big data environments and outside enterprise systems without any duplicative cost is essential.
- *Multi Tenancy*: A multi-tenant network which can consolidate and centralize the big data management.
- *Network Intelligence*: Network provisioning using sophisticated intelligence to handle workloads as per the requirements or priorities.
- *Network Partitioning*: This is one of the most significant features as different organizations can require separate network partitions for running their tasks. For example, a research organization can run multiple projects which need to be separated from regular transactions.
- *Scale Out*: The ease of scalability and seamless transitions in terms of size and number need efficient network management to reduce over subscriptions.

However, conventional network infrastructures provide rigid network polices for FinTech applications. The control and data planes are collocated to each other making the policies and protocols very restrictive in nature. This means in the traditional suite, the software/control logic is embedded within the networking devices such as-switches and routers. Thus, the individual configuration of such network devices is a challenging task due to their high maintenance cost. Moreover, in case of relocation of hardware devices, a complex procedure is used to reconfigure the network itself [16]. TCP/IP model has many other issues such as-global monitoring and flow control, logical grouping, manual configuration, policy modification, and hardware migration. Hence, the traditional network paradigm has to face numerous challenges to meet the requirements in multi edge-cloud ecosystem serving the Fintech. However, in a FinTech environment, one of the most important prerequisites is a flexible and adaptive network architecture which could handle the incoming data traffic in an robust manner.

So, to mitigate the aforementioned issues and to provide scalability, reliability, and programmability to the network architectures, software-defined network (SDN) has evolved as one of the most prominent technologies over the years. Many researchers have explored the use of SDN in edge-cloud ecosystem from various aspects of cost, delay, carbon emission, and energy [17]. SDN has a centralized architecture that: (1) decouples the network control from forwarding devices, and (2) provides abstraction at control plane. To control the network topology, SDN depends on the central intelligence in comparison to legacy networks. The central controller handles the abstraction of network complexities from application developers using northbound application programming interfaces (APIs) and deploy network functionalities into forwarding devices using southbound APIs. However, these centralized controllers may act as a single point of failure, hence a fault-tolerant and logically distributed multi-controller setup becomes necessary for the scalability of the network over a large area [18].

B. Motivation and Research Challenges

With the increase in usage of edge and cloud computing for different application domains, the dynamic traffic flow management has become a tedious task. Moreover, the number of switches is increasing drastically in multi edge-cloud ecosystem which puts additional burden on network resources with respect to flow control, reliability, and energy-efficiency. For this purpose, there is an imminent need of an efficient solution which helps to manage the overall network traffic in an optimal manner. These issues are more prominent in traditional

TCP/IP networks, therefore these networks are being replaced by more adaptive network architectures such as SDN for providing efficient services in domains such as-healthcare [19], smart grid [20], smart cities [21], vehicular networks [22], and data center management [23]. However, the major issue in SDN-enabled network architectures is the use of a single controller at a central location. This may lead to reliability and fault tolerance issues. But, with recent advancements in SDN, multiple controllers and virtualization of physical controllers is possible which helps to overcome these drawbacks. However, the optimal number of controllers and their placement in the underlying network is a very important task. This is because, if there are fewer controllers, then it would lead to more delay and less reliability; if there are more controllers, then it would lead to more energy consumption and under-utilization of network resources. Moreover, to cater to the ever increasing requirements of providing seamless services in the underlying networks, the placement of controller plays a crucial role in addressing various key issues such as-reliability, latency and energy-efficiency. Thus, the primary concern in SDN architectures is handling *Controller Placement (CP)* problem in order to address the end-users requirements (with respect to latency and reliability).

CP problem has been addressed by many existing proposals specifically for Data center networks [24], Large Scale SDN Networks [25], Software-defined wide area networks [26], Wireless Edge Networks [27]. In [24], the authors used a coalition formation games to form different clusters for stable network partitioning and localization of controllers. In [25], the authors used Pareto-based Optimal COntroller placement approach based on several performance metrics. In [26], the authors considered the switch-controller/inter-controller latency requirements and the capacity of the controllers to realise the resilient controller placement using clique-based approach in graph theory. In [25], the authors used linearization and supermodular function techniques to model the CP problem that lead to approximate solutions which outperforms the state-of-the-art methods in performance. However, all these proposals fail to address the following questions:

- Q1: What happens if a controller fails in a multi-controller setup?
- Q2: Do these existing techniques for controller placement dynamically select an alternative controller if a running controller fails?
- Q3: What conditions or performance parameters should be considered before the switching between multiple controller in a failure scenario.

Although some proposals have successfully proposed failure recovery or fault tolerant mechanism for SDN setup, but most of them are focused on a single controller scenarios. For a multi-controller or distributed scenario, the authors in [28] designed a self-stabilizing placement mechanism for the migration capable controller instances in IoT network. In [29], the authors proposed a heuristic algorithm to address the controller placement problem with respect to single-link or multi-link failures. In the above proposals, there is no consideration of a multi-controller scenario where the network load in divided across different network partitions. In [30], the authors considered a master and slave architecture to resolve the problem of failure but this may not adapt to large scale network flows where the requirements are very dynamic. This also limits the scalability of the overall distributed setup. This problem was resolved to a limited extend in [31] where they used multiple slave controller. However, they do not consider dynamic switching based on different performance parameters.

In summary, these above raised challenges for SDN controller failure need to be resolved using an adaptive recovery mechanism that considers different parameters (like reliability, latency, and energy) while considering the switching from a failed controller to an active controller. Looking into the FinTech applications that are distributed across the globe, it become pretty necessary to devise a recovery mechanism before utilising SDN in there setup. For example, the financial banking sector has several branches located at different geo-location across various countries. So, if a SDN controller handling the network traffic related to these FinTech applications fail, then there may be a possibility that entire segment or area goes down or remains offline and the customers may not access the service or may access limited services. This is not a very preferable scenarios for FinTech applications as any failure may result in faulty transactions and even attackers or malicious parties can take advantage of such scenarios.

C. Contributions

Keeping in view of the above considerations, We propose an Adaptive Recovery Mechanism for Multi-controller scenario for SDN applicable for FinTech Applications. The major contributions are as follows.

- For a large-scale edge-cloud ecosystem supporting FinTech, a adaptive recovery mechanism has been proposed for failure recovery in multi-controller SDN scenario.
- A controller classification scheme is designed based on Support Vector Machine algorithm that decides which controllers lie in the category of tentative load bearers for a failed controller.
- The evaluation of the proposed scheme is carried out on different competing objectives (reliability, energy consumption, and latency) using a case study of typical city.

II. Network Model and Preliminaries

A. Network Model

We represent the SDN network using an undirected weighted graph G = (V, E); where V depicts the set of openflow switches and controllers and E represents the set of links/edges between the pair of switches or controllers. In this network, each edge is associated with a definite weight which represents the distance ($d_{i,j}$) between i^{th} and j^{th} pair of switches (s) or controllers (c). The undirected weighted graph of a sub-network z is denoted using $G^z = (V^z, E^z) \mid V^z \subset V, E^z \subset E$; where the associated set of vertices and edges are represented using V^z and E^z.

B. Assumptions

The control plane consists of identical SDN controllers. Some of the assumptions and initial conditions for the proposed model are given as follows:

- Initially, all the controller are switched on.
- The control plane system is down when all the SDN controllers are in a failed condition.
- The control plane system has failure when one of more SDN controllers are in a failed condition.
- Switching is perfect and instantaneous.
- Distribution of failure time is taken as negative exponential while the recovery time it is considered as arbitrary.

C. System States

The system states for the failure-recovery mechanism are shown in Fig. 2 and the description is provided as below.

- **Good State** (S_o): In this state, all the controllers are in working condition. Some of the operative controllers are added to a standby list selected by proposed scheme to act as backups in case of failure.
- **Failure State** (S_1): In this state, one controller fail and a part of network goes down. Here, the failure rate (λ) play important role.
- **Transition State** (S_2): In this state, the control of the failed SDN controller is automatically transited to another controller selected from the standby list.

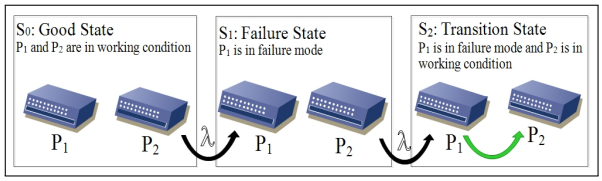

Fig. 2. Various States of System (for simplicity we depicted a 2 controller scenario but P_2 is selected from standby list); λ: failure rate

Various states of the proposed system are shown in Table I.

TABLE I
STATE-SPECIFICATIONS

State	Specification
S_0	One controller is operative and the others are in standby list
S_1	One controller has failed and the other controllers are operative
S_2	Recovery from state S_1 through transition from failed controller to an operative controller from standby list.

D. State Transition Analysis

In this paper, initially the system is assumed to be in state S_0, in which all the controllers are switched ON and the proposed scheme selects some of the operative controller and add them to a standby list. When any operative controller fails, the system transits from state S_0 to state S_1. In this state, a standby controller selected using the proposed scheme becomes operative and the system transits from state S_1 to state S_2. It is assumed that switching is perfect and instantaneous. Once the failed controller is recovered, it is added to the standby list and the system return back to state S_0.

III. Proposed Scheme

The proposed scheme works in two phases one after another. In the first phase, an SVM-based classification approach is proposed that separates the controllers that can act as standby to recover a failed controller from others. In the second phase, an appropriate controller is selected from the standby list received from the first phase based on multiple parameters (reliability, energy, and latency). These phases are described comprehensively in the followings sections. The switches and controllers are the vertices and the path between the pair of switches or controllers are the edges in the network graph.

A. SVM-based Classification Approach

The SVM-based classification approach (hidden layer SVM) is used to classify the switches and controllers into two lists, i.e., standby and non-standby. The standby list consists of the controllers or switches that can considered for backup purpose to recover the consequences of any failed controller or a switch. The non-standby list stored the detail of the available controllers or switches except the one that failed. The failed controllers are automatically added to a new list and these controllers are repaired and further they are ready to operate as per the requirement. SVM approach is based on supervised machine learning technique to differentiate the overall group into pre-defined classes using different decision functions. For this purpose, the model is trained using processed data by considering the required features or parameter (such as post number, health status, network load, communication cost, error rate, failure rate, etc) and thereafter the testing approach is performed to validate the preciseness of the model. SVM approach create a hyper plane to differentiate the classes with proper margins.

SVM consists of different layers, i.e., input layer, feature layer and output layer. These layers are shown in Fig. 3. A bias (b) is used to help the activation function to select the optimal class. Here, an input vector (n) is forwarded to the input layer of the SVM model for classification. There are various training neurons $(m_1, m_2, .., m_i, .., m_q)$ on the Input layer and $\mu \in (0, 1)$ is the vector to store the classification output [32], [33]. The trained instances are matched with the input vector n by using specified activation function (K) in the feature layer [34]. To define the boundaries of the hyper plane, the weight vector (w) and bias value b is used as follows.

$$w.n + b = 0 \tag{1}$$

The function to classify the training and testing is given as below.

$$f(n) = sign(w.n + b) \tag{2}$$

During mapping the neurons and the input vector, the following kernel function is used.

$$f(n) = sign(\sum_{i=1}^{N} \alpha_i \mu_i K(m_i, n + b) \tag{3}$$

where, K represents the kernel function to map the neuron features $(m_1, m_2, .., m_i, .., m_q)$ with the input vector (n).

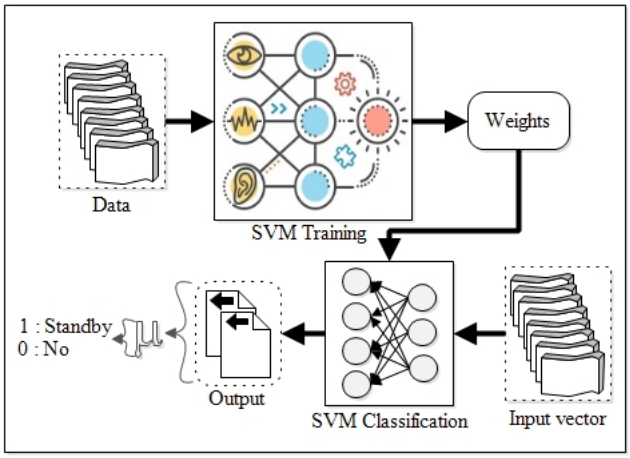

Fig. 3. SVM classification

Now, the objective function to define the boundaries of the hyper plane is defined as below.

$$f(O) = min(\frac{1}{2}w^T w) + C\sum_i max(0, 1 - \mu_i w^T m_i) \quad (4)$$

where, $min(\frac{1}{2}w^T w)$ is the regularization term for better generalization of the hypothesis spaces, C is the controller to define the high margins between the boundaries and $max(0, 1 - \mu_i w^T m_i)$ is the empirical loss to penalize the weight vectors for inaccurate selection of boundaries.

During classification, an optimized function is used to categorize the incoming input vector n, as mentioned in Eq. 5.

$$\mu = \sum_i^N \alpha_i K(m_i, n) + b \quad (5)$$

The steps are followed during the training and testing the defined model are described as below [35]:

- Select the required parameters for training the model as per network requirements.
- Divide the dataset into two parts, one chunk is used for the training purpose and other one is for testing purpose.
- Increment the epoch value and re-run the model again and again till the required accuracy is not achieved.
- During increment in epoch values, the weights of the neurons are updated and kernel trick technique is used for efficient weight allocation at different layers.
- When the model is fully trained, then configure the network setup to validate the results accordingly.

The entire classification process is presented in Algorithm 1.

Initially, the random weights are assigned to the neurons of the layers. Then, the forwarded data is mapped with the trained instances by using the features and mapped accordingly to the particular class. After on-wards, the margin between the boundaries of the defined class is calculated known as hyper plane. For clarification, objective function is calculated to define the boundaries more precisely. If, the calculated margin is not justified, then update the weights again and recalculate the objective function. Lastly, the output of the proposed algorithm is mentioned below:

Algorithm 1 SVM-based classification

Input: $n = (n_1, n_2, ..., n_J)$
Output: μ

```
Initialize weights W^0 for each layer.
for j=1 : J do
    for α Iterations do
        for i=1 : N do
            Calculate the initial boundary distance.
            Calculate the objective function f(O).
            Check the boundary margins: D_b = 2/||w||
            Compare D_b with expected margin
            Otherwise: Iterate α.
            Repeat steps 5-7 till α → μ
        end for
    end for
```

$$\mu = \begin{cases} 1: & \text{Standby} \\ 0: & \text{No} \end{cases} \quad (6)$$

B. Recovery Controller Selection Scheme

The considered problem deals with the selection of a suitable neighbourhood switches and controllers in the given network, based on three parameters, i.e., latency, reliability, and energy. These parameters are considered as objective functions that must be achieved for selecting a suitable controller/link for recovery and accordingly rebuild the entire recovery network. The formal description of these objective functions is given as follows:

*1) **Objective Function 1 (O_1)** : $Reliability$:* The network reliability is one of the key aspect as it determines the resilience of the controllers. The controllers and the number of nodes directly connected to them are major factors for selecting the reliable controller. This estimation can be performed using a reliability index ($\mathbf{R}(i,n)$) of each controller in the network. The $\mathbf{R}(i,n)$ for i^{th} switches connected to n^{th} controller can be computed as follows [36]:

$$\mathbf{R}(i,n) = \frac{1}{|L^n|}\left(T^{(DW)}(v_{i,n}) \times \lambda_{i,n}\right); \forall i, n \quad (7)$$

where, $|L^n|$ represents the total number of switch-controller links in the network, $N^{DT}(v_{i,n})$ represents the number of edges that are experiencing the downtime, and $\lambda_{i,n}$ denotes the average failure rate of the switch-controller link $v_{i,n}$.

*2) **Objective Function 2 (O_2)** : $Energy$:* The total energy consumption must be considered while selecting a controller where the network load of a failed controller will be shed as otherwise this may lead to the destabilisation or overloading of the entire network. The energy consumption (E(i,n)) is defined as below.

$$\mathbf{E}(i,n) = \times\left[\left(E^{FX}_{i,n} \times (t^{en}_{i,n} - t^{st}_{i,n})\right) + \left(E^{DY}_{i,n} \times (t^{en}_{i,n} - t^{st}_{i,n}) \times |\mathcal{N}^{AP}(v_{i,n})|\right)\right]; \forall i, n \quad (8)$$

where, $E_{i,n}^{FX}$ represents the fixed part of energy consumption (fans, chassis, etc), $E_{i,n}^{DY}$ is the dynamic part of energy consumption that depends on the number of active ports ($|\mathcal{N}^{AP}(v_{i,n})|$), $t_{i,n}^{en}$ and $t_{i,n}^{st}$ denotes the start and end times of i^{th} switch [37].

*3) **Objective Function 3 (O_3)** : $Latency$:* Latency plays a key role in selecting the recovery controller in the network. The latency $\mathbf{T}(i,n)$ in the network is defined as below.

$$\mathbf{T}(i,n1,n2) = \left[\left(\frac{D_{i,m,n}}{T_{PROP}}\right) + \left(T_{i,n}^{PROC}(t)\right) + \left(\frac{\mathcal{P}_{i,n,j}^{SZ}(t)}{B_{i,n,j} \times O_{i,n,j}^{RT}(t)}\right) + \left(\frac{|Q_{READY}(t)|}{B_{i,n,j} \times O_{i,n,j}^{RT}(t)}\right)\right]; \forall i,n,j \quad (9)$$

where, $D_{i,m,n}$ depicts the weighted distance between the pair of switches ($d_{i,m,n}$), T_{PROP} the medium propagation delay, $T_{i,n}^{PROC}(t)$ represents the processing delay, $mathcalP_{i,n,j}^{SZ}(t)$ denote the packet size at time instant t, $B_{i,n,j}$ denotes bandwidth, $O_{i,n,j}^{RT}(t)$ is the occupation ratio of the j^{th} port of the i^{th} switch, and $Q_{READY}(t)$ is depth of the ready queue.

C. Problem formulation

In order to select a suitable controller to handle the network load of a failed controller in multi edge-cloud environment, the above defined objective functions are considered to design a combined utility function matrix. The combined utility matrix is given as below.

$$\hat{U}_{i,j,n}^{map} = \sum_{i=1}^{k} \begin{bmatrix} 1,1,1 & 1,2,1 & . & . & 1,j,1 \\ 1,1,2 & 1,2,2 & . & . & 1,j,2 \\ . & . & . & . & . \\ . & . & . & . & . \\ 1,1,n & 1,2,n & . & . & 1,j,n \end{bmatrix} \quad (10)$$

Now, to select the optimal mapping for all the three objective functions, a decision variable is defined as below.

$$\alpha_{(i,j,n)} = \begin{cases} 1 & \text{for if all three objectives are achieved} \\ 0 & \text{for } otherwise \end{cases} \quad (11)$$

This combined utility function is formulated using integer linear programming and is given as below.

$$max \left[\sum_{j=1}^{j_n} (_{1j_1}1)\alpha_{ij} +_{1j_2 2} \alpha_{1j_2 2} + ... +_{1j_k n} \alpha_{1j_n k}\right] \quad (12)$$

subject to following constraints

$$\alpha_{ijn} \in [0,1]$$
$$\mathbf{R}(i,n) > 0$$
$$\mathbf{E}(i,n) > 0$$

Let $\mathbf{T}_{exp}$ represent the expected or worst bound delay for each switch/vertex in the considered network, G ={V,E}. Now, we define a recovery neighborhood (RN) of switch *i*, as below.

$$\mathbf{RN}(i) = \{i | j \in \mathbf{V}\};\ s.t.\ \ d_{i,j} < \mathbf{T}_{exp} \quad (13)$$

where, $d_{i,j}$ denotes the smallest delay/latency between the switches *i* and *j*.

Algorithm 2 Recovery neighborhood selection algorithm

Input: $G = (V,E), T(i,m,n)_{thr}, R(i,n)_{thr}$
Output: N

```
Compute Delay Matrix using Floyds Algorithm.
Initialize N as an Identity matrix.
for i ← V do
    Find N(i) in order of delay to i.
    N(i) ← i
    if R^c(G^c [N(i) U {j}] < R_0^c then
        N(i) ← V(i) U {j}
        n(ij) = 1
    end if
    for i ← n(ij) do
        if E^c(G^c [N(i) U {j}] < E_0^c then
            α(ijk) = 1
            N
        end if
    end for
```

This depicts $\mathbf{RN}(i)$ as a cluster of switches that can be linked with *i* within the worst bound delay. This means that if *i* is a controller then all the switches connected to it must belong to $\mathbf{RN}(i)$. This helps to ensure that the objective of delay is satisfied. Hence, the first task is to create a recovery neighbourhood from the class of switches/controllers that has been added into the standby list.

We proposed the Algorithm 2 to show how the recovery neighbourhood mechanism works in line with the three objectives. For a given network, a recovery neighbourhood network is created that meets the three objectives while recovering the failed controller and switches/links that are down as a consequence of the failure. We can say that if switch *j* belongs to the neighbourhood of *i*, then n_{ij} = 1, otherwise n_{ij} = 0. In the proposed algorithm, initially a delay matrix is constructed using the Floyd's Algorithm [38]. Now, the network N is initialised as an identity matrix. After this, for each switch *i*, we construct RN(i). Using RN(i), we compute N(i) by combining the switch/vertex that satisfies the given constraint. If $k = |\mathbf{V}|$, then in Y = $(y_0, y_1, ..., y_{k-1})$, if y_i = 1 then it means the controller resides at vertex *i*, otherwise y_i = 0. Now, our target is to select the minimum number of controller where the failed part of network should be shifted so that all the objectives are satisfied. The selected controller should cover the entire network N with maximum reliability and minimum delay as well as energy consumption. For this reason, we have defined the following objective function.

$$min : \sum_{i=0}^{k-1} y_k \quad (14)$$

such that

$$C1 : \sum_{i=0}^{k-1} y_i n_{ij} > 0$$
$$C2 : 0 \leq j \leq k$$

Using the above objective function, we can select the minimum distanced switch-controller links and assign them as control path, i.e., $\mathbf{G}_c(\mathbf{Y}_i)$. In the next step, we calculate the reliability index ($\mathbf{R}_c$) using Eq. 7. The $\mathbf{R}_c$ is compared for all controllers in $\mathbf{G}_c(\mathbf{Y}_i)$ to select the controllers that achieve minimum delay with maximum reliability. Finally, to achieve the objective of minimum energy consumption, we compute the energy consumption of switches and controller using Eq. 8. Once the thrid objcetive is achieved, we set α_{ijk} = 1. This way, we are able to create a complete recovery network (including controller, switch, links, etc) to divert the network traffic from failed part of network.

IV. Results and Discussion

To test the efficacy of the proposed scheme, a typical city network has been considered as a case study. The proposed scheme is evaluated in terms of reliability, energy consumption, and latency.

A. Case study of a typical City

A realistic example of a metropolitan area network, i.e., a typical city has been considered for evaluation of the proposed scheme. In this example, the underlying SDN network in multi edge-cloud ecosystem have some locations with dense network of switches and others having sparse network. In the multi edge-cloud network, 5 distribution switches are considered to be placed in different locations which are further connected to 100 core/edge switches that are deployed throughout the network. The entire network has been divided into different network clusters. We have considered four scenarios for evaluation with different numbers of controller set, i.e., c: {2,4,6,8}. In each scenario, we consider that one of the controller fail and the proposed scheme puts the remaining controllers along with associated switches into two list, i.e., standby and non-standby using SVM as discussed in Section III-A. From this list, our scheme selects a suitable controller and switches where the load of failed controller can be shed. For SVM training, the dataset has been divided in 70-30 ratio where 70% is used for training and the rest is used for testing.

Initially, the reliability analysis of each distribution switches connected to core and edge switches is performed with respect to the failure rates. For this purpose, the considered case of disconnected switches and associated links for failed network is shown in Fig. 4. Further, the Fig. 5 shows failure rate with respect to various switches collected from the city network. Fig. 6 shows the reliability index of the switches with respect to the failure rates. For analysis of variation of reliability index for a change in the value of failure rate, four categories of failure rate $\lambda_{i,n} = (0.25, 0.5, 0.75, 1.0)$ are considered. Fig. 7 shows the effect of variation in failure rate on the reliability index.

The impact of number of controllers and flows is examined with respect to latency. Fig. 8 shows that C=6,8 are having lower latency in comparison to C=2,4. In this case, the latency of C=6 is marginally lower than C=8. So, if we analyze all these parameters, then an optimal solution is achieved with C=6 as the variation in latency is marginally higher than C=8,

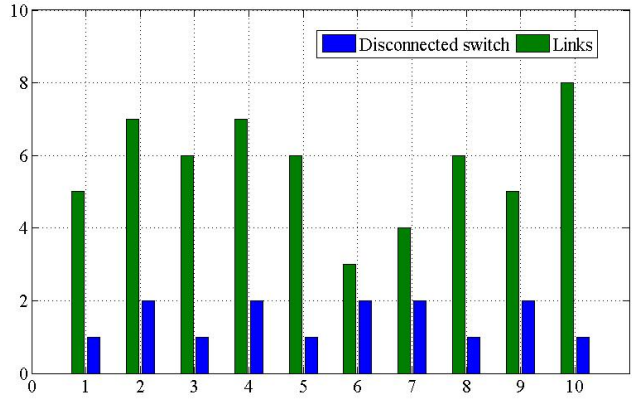

Fig. 4. Disconnected switches and links in failure.

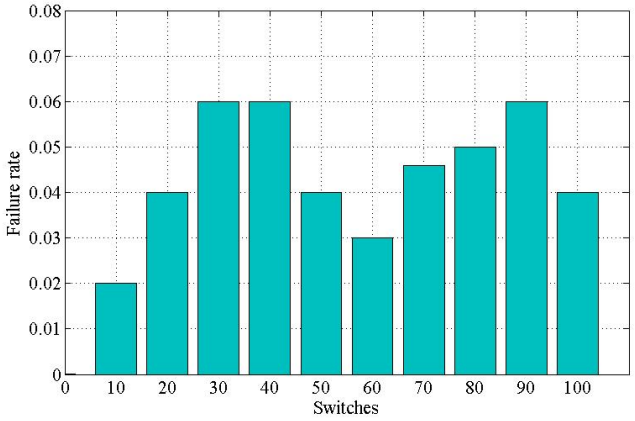

Fig. 5. Failure rate with respect to number of switches.

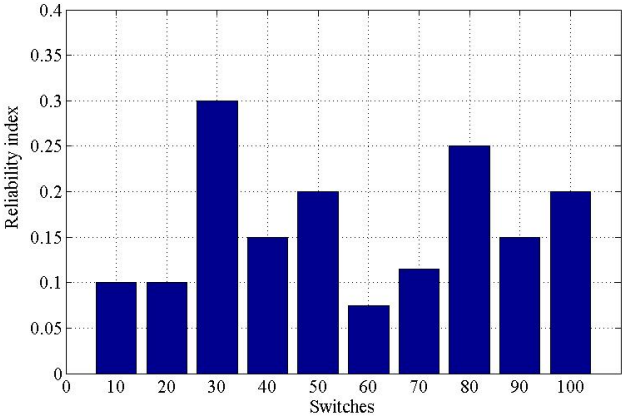

Fig. 6. Variation of reliability index.

but the energy consumption is lowest when compared with other values of C.

Moreover, in a multi edge-cloud ecosystem, energy consumption of the controllers for providing seamless services is an important aspect. In this regard, the variation of energy consumption using the proposed scheme with increasing number of controllers and flows are analyzed. Fig. 9 depicts that the energy consumption with C=6 is the lowest. This is due to the

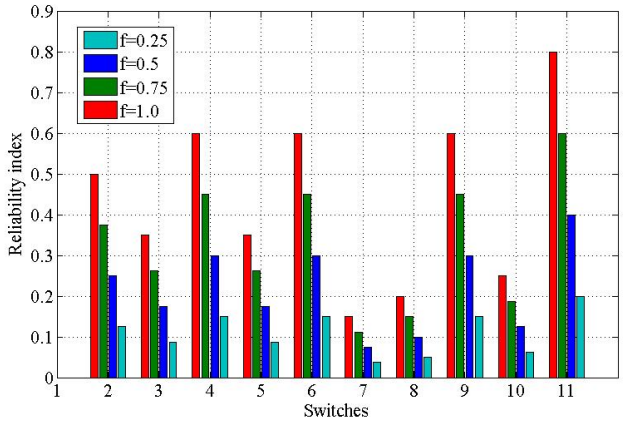

Fig. 7. Reliability index vs number of flows.

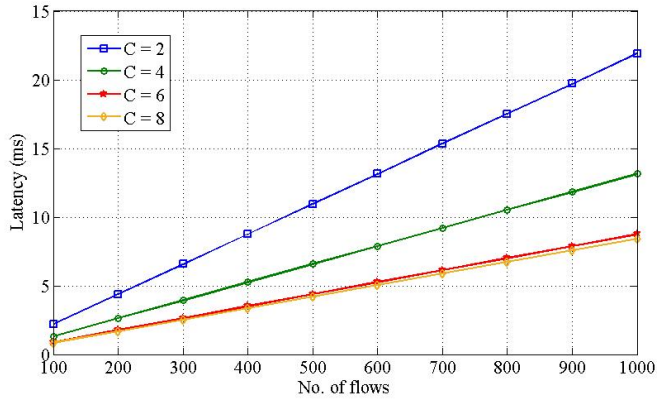

Fig. 8. Latency with respect to number of flows.

reason that with lower number of controllers, the load on each switch increases which maximizes the energy consumption. However, for C=8, the energy consumption is somewhat higher than C=6, as in this case the number of controllers exceeds the network load. So, an additional controller increases the overall energy consumption in the network.

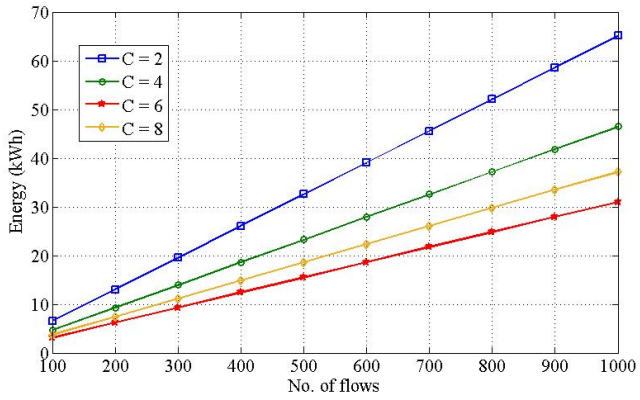

Fig. 9. Energy with respect to number of flows.

V. Conclusion

SDN has evolved as a major technology in the modern era for the purpose of traffic monitoring, flow management, scalability, and flexibility. However, dependency on a single centralized controller may lead to the inefficiencies and bottlenecks for any implemented solution for various applications in SDN. Even if, this problem is resolved using multiple controller, there come a challenge when one of the controller fails. In such a case, it become a necessity to provide a mechanism to select a suitable controller that can act as backup for the recovery of the portion of the network that goes down. The proposed scheme addresses this challenge to provide efficient solution with respect to different parameters such as-latency, energy consumption, and reliability. The results shows that the proposed scheme behaves as desired for all the three parameters to select the recovery neighbourhood.

Acknowledgement

This work is partially supported by the Durham University Startup Fund and Major Scientific and Technological Projects of CNPC under Grant ZD2019-183-006, partially supported by Shandong Provincial Natural Science Foundation under Grant ZR2020MF006, and partially supported by ”the Fundamental Research Funds for the Central Universities” of China University of Petroleum (East China) under Grant 20CX05017A.

References

[1] J. Kagan, *Financial Technology – Fintech*, Aug 2020. [Online]. Available: https://www.investopedia.com/terms/f/fintech.asp

[2] *Fintech (Financial Technology)*, Aug 2020. [Online]. Available: https://corporatefinanceinstitute.com/resources/knowledge/finance/fintech-financial-technology/

[3] N. Kumar, T. Dhand, A. Jindal, G. S. Aujla, H. Cao, and L. Yang, “An edge-fog computing framework for cloud of things in vehicle to grid environment,” in *2020 IEEE 21st International Symposium on” A World of Wireless, Mobile and Multimedia Networks”(WoWMoM)*. IEEE, 2020, pp. 354–359.

[4] R. R. Suryono, I. Budi, and B. Purwandari, “Challenges and trends of financial technology (fintech): A systematic literature review,” *Information*, vol. 11, no. 12, p. 590, 2020.

[5] P. Zhang, C. Jiang, X. Pang, and Y. Qian, “Stec-iot: A security tactic by virtualizing edge computing on iot,” *IEEE Internet of Things Journal*, 2020.

[6] P. Zhang, X. Pang, N. Kumar, G. S. Aujla, and H. Cao, “A reliable data-transmission mechanism using blockchain in edge computing scenarios,” *IEEE Internet of Things Journal*, 2020.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[8] P. Zhang, G. Singh, N. Kumar, and M. Guizani, “Iov scenario: Bandwidth aware algorithm in wireless network communication mode,” *IEEE Transactions on Vehicular Technology*, 2020.

[9] P. Zhang, X. Pang, Y. Bi, H. Yao, H. Pan, and N. Kumar, “Dscd: Delay sensitive cross-domain virtual network embedding algorithm,” *IEEE Transactions on Network Science and Engineering*, 2020.

[10] P. Zhang, C. Wang, C. Jiang, and A. Benslimane, “Security-aware virtual network embedding algorithm based on reinforcement learning,” *IEEE Transactions on Network Science and Engineering*, 2020.

[11] J. Liu, X. Kang, C. Dong, and F. Zhang, “Simulation of real-time path planning for large-scale transportation network using parallel computation,” *INTELLIGENT AUTOMATION AND SOFT COMPUTING*, vol. 25, no. 1, pp. 65–77, 2019.

[12] M. Liu, X. Zhang, S. Ge, X. Chen, J. Wu, and M. Tian, “An application-oriented buffer management strategy in opportunistic networks,” *CMC-COMPUTERS MATERIALS & CONTINUA*, vol. 60, no. 2, pp. 559–574, 2019.

[13] M. Okhovvat and M. R. Kangavari, "Tslbs: A time-sensitive and load balanced scheduling approach to wireless sensor actor networks," *COMPUTER SYSTEMS SCIENCE AND ENGINEERING*, vol. 34, no. 1, pp. 13–21, 2019.
[14] J. Wang, Y. Gao, W. Liu, W. Wu, and S.-J. Lim, "An asynchronous clustering and mobile data gathering schema based on timer mechanism in wireless sensor networks," *Comput. Mater. Contin*, vol. 58, no. 3, pp. 711–725, 2019.
[15] N. Al-Otaiby, H. Kurdi, and S. Al-Megren, "A hierarchical trust model for peer-to-peer networks," *CMC-COMPUTERS MATERIALS & CONTINUA*, vol. 59, no. 2, pp. 397–404, 2019.
[16] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on computing, network, and storage resource constraints," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3298–3304, 2017.
[17] P. Borylo, A. Lason, J. Rzasa, A. Szymanski, and A. Jajszczyk, "Energy-aware fog and cloud interplay supported by wide area software defined networking," in *IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–7.
[18] M. T. I. ul Huque, G. Jourjon, and V. Gramoli, "Revisiting the controller placement problem," in *IEEE 40th Conference on Local Computer Networks (LCN)*, 2015, pp. 450–453.
[19] L. Hu, M. Qiu, J. Song, M. S. Hossain, and A. Ghoneim, "Software defined healthcare networks," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 67–75, 2015.
[20] S. Rinaldi, P. Ferrari, D. Brandão, and S. Sulis, "Software defined networking applied to the heterogeneous infrastructure of smart grid," in *IEEE World Conference on Factory Communication Systems (WFCS)*. IEEE, 2015, pp. 1–4.
[21] J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin, "Software-defined internet of things for smart urban sensing," *IEEE communications magazine*, vol. 53, no. 9, pp. 55–63, 2015.
[22] G. S. Aujla, R. Chaudhary, N. Kumar, J. J. P. C. Rodrigues, and A. Vinel, "Data offloading in 5G-enabled software-defined vehicular networks: A stackelberg game-based approach," *IEEE Communication Magazine*, vol. 55, no. 8, pp. 100–108, Aug 2017.
[23] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving energy in data center networks." in *NSDI*, vol. 10, 2010, pp. 249–264.
[24] R. Chaudhary and N. Kumar, "Parc: Placement availability resilient controller scheme for software-defined datacenters," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8985–9001, 2020.
[25] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.
[26] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined wans," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, 2018.
[27] Q. Qin, K. Poularakis, G. Iosifidis, S. Kompella, and L. Tassiulas, "Sdn controller placement with delay-overhead balancing in wireless edge networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1446–1459, 2018.
[28] S. Chattopadhyay, S. Chatterjee, S. Nandi, and S. Chakraborty, "Aloe: Fault-tolerant network management and orchestration framework for iot applications," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2020.
[29] S. Yang, L. Cui, Z. Chen, and W. Xiao, "An efficient approach to robust sdn controller placement for security," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1669–1682, 2020.
[30] L. Sidki, Y. Ben-Shimol, and A. Sadovski, "Fault tolerant mechanisms for sdn controllers," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016, pp. 173–178.
[31] A. Mahjoubi, O. Zeynalpour, B. Eslami, and N. Yazdani, "Lbft: Load balancing and fault tolerance in distributed controllers," in *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, 2019, pp. 1–6.
[32] L. J. Herrera, I. Rojas, H. Pomares, A. Guillén, O. Valenzuela, and O. Baños, "Classification of mri images for alzheimer's disease detection," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 846–851.
[33] M. Pal and G. M. Foody, "Feature selection for classification of hyperspectral data by svm," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 5, pp. 2297–2307, 2010.
[34] S. Bauer, S. Köhler, K. Doll, and U. Brunsmann, "Fpga-gpu architecture for kernel svm pedestrian detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE, 2010, pp. 61–68.
[35] G. M. Foody and A. Mathur, "Toward intelligent training of supervised image classifications: directing training data acquisition for svm classification," *Remote Sensing of Environment*, vol. 93, no. 1-2, pp. 107–117, 2004.
[36] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in SDN," in *IEEE/IFIP Network Operations and Management Symposium (NOMS),*, 2016, pp. 481–487.
[37] G. S. Aujla and N. Kumar, "SDN-based energy management scheme for sustainability of data centers: An analysis on renewable energy sources and electric vehicles participation," *Journal of Parallel and Distributed Computing*, 2017, doi:10.1016/j.jpdc.2017.07.002.
[38] B. Xue, Y. Guiqin, and J. Zhanjun, "Research and application of floyd algorithm based on sdn network," in *2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, 2019, pp. 317–320.