Proactive Controller Assignment Schemes in SDN For Fast Recovery

Selcan Güner
Dept. of Computer Engineering
Bogazici University
Istanbul 34342, Turkey
selcan.guner@boun.edu.tr

Gürkan Gür
Zurich University of Applied Sciences (ZHAW)
Inst. of Applied Information Technology
Winterthur 8401, Switzerland
gueu@zhaw.ch

Fatih Alagöz
Dept. of Computer Engineering
Bogazici University
Istanbul 34342, Turkey
fatih.alagoz@boun.edu.tr

***Abstract*—A sizeable software defined network with a single controller responsible for all forwarding elements is potentially failure-prone and inadequate for dynamic network loads. To this end, having multiple controllers improves resilience and distributes network control overhead. However, when there is a disruption in the control plane, a rapid and performant controller-switch assignment is critical, which is a challenging technical question. In this work, we propose a proactive switch assignment approach in case of controller failures using a genetic algorithm based heuristic that considers controller load distribution, reassignment cost and probability of failure. Moreover, we compare the performance of our scheme with random and greedy algorithms. Experiment results show that our proposed PREF-CP framework has better performance in terms of probability of failure and controller load distribution.**

I. Introduction

Conventional computer networks consist of network elements such as switches and routers with many complicated protocols, policies and communication interfaces. Their structure is restrictive for network operators to change the network according to the needs of changing traffic demands and service requirements. This shortcoming is becoming more taxing with the increasing number of mobile devices and impact of big data flows [1], [2]. Furthermore, the proliferation of advanced wireless systems such as 5G networks and massive connectivity of IoT impose stringent performance requirements on the wired communication infrastructure such as backhaul and core networks [3]. The idea of Software Defined Networking (SDN) was proposed to facilitate network evolution while addressing these challenges to realize the Future Internet concept [4]. With SDN, control decisions and network intelligence is moved out of individual network nodes, which transforms the network to simpler forwarding hardware augmented with decision making network controllers.

For various software-defined systems, a single controller might be sufficient since a single controller may meet the service level requirements under specific conditions [5]. However, resilience is an important aspect when designing a network architecture. As the system is functioning, failures can occur in both control and forwarding planes. A switch, a controller or links between them may fail. For instance, when a switch is disconnected from its controllers, it can not forward new flows and becomes unresponsive except residual flows in time. A failure may eventually cause loss of data which reduces the reliability of the system. Therefore, it is of great importance to improve resilience of software-defined networks [2]. In a network with a single controller, when that controller fails, the network will be left without a control framework, i.e. become "headless". To overcome this single point of failure problem, control plane is distributed over multiple controllers to increase resilience and simultaneously provide adaptation to dynamic network loads [6]. The distribution of switches to controllers affects multiple aspects of a network such as controller-switch latency, load balancing and network reliability. Therefore, with use of multiple controllers, the dynamic and responsive controller-to-switch assignment becomes crucial.

In this work, we focus on this problem and investigate the efficient controller-to-switch assignment problem in SDN to overcome effects of failures. We propose ***P**roactive **RE**covery **F**ramework for SDN **C**ontrol **P**lane* (PREF-CP) which is a proactive switch assignment scheme against controller failures via genetic algorithm considering controller load distribution, reassignment cost and probability of failure. The main premise is to have a pre-calculated mapping for the network calculated at run-time and applied rapidly when failure incidents happen.

II. Related Work

Controller failure recovery mechanisms in the literature typically consider control plane reliability of the network, switch-controller delay, or controller load. When calculating controller-switch assignment for multiple controller environment in SDN, multiple parameters such as controller load, probability of failure between network components or reassignment cost are considered [7]. For the controller load aspect, Müller et al. propose a reassignment algorithm which chooses a controller with highest controller capacity in [8]. For considering switch-controller delay, Obadia et al. propose a failure recovery mechanism where the nearest controller takes over the switches which were under control by a failed controller [6]. As studied in [8], [9], controller capacity and controller overload are also critical issues for switch assignment planning.

To assure the reliability of the control plane, the most important aspect is to keep switches connected to controllers which are up and running [10]. In that regard, one important

metric is probability of failure which is defined as likelihood of communications failure from *node_a* to *node_b* [11]. Another important requirement is related to recovery speed after failures: to minimize disruptions in case of controller failure, rapid reactions are needed with failover mechanism(s) ensuring connectivity with remaining components [6]. To prevent long backup calculation and restoration time, [12], [6] and [9] generate a backup map proactively. There are also works which calculate reassignment on the fly when a failure occurs [8]. Depending on the backup calculation time, both approaches have pros and cons. Proactive approaches may be faster to recover from a failure. However, calculation on the fly may use live network data while proactive calculations will use the last state of the system when the decision algorithm was run.

III. Proactive SDN Controller-Switch Assignment (PREF-CP)

Software or hardware malfunction in the machine hosting the controller can cause failures on control plane. There are different assignment options that can be used in case of controller failure at runtime. For instance, the switches of failed controller(s) can be randomly assigned to other controllers. However, such a strategy does not provide satisfactory performance [7]. Optimizing controller-to-switch assignment problem is NP-hard, so it is crucial to find an efficient assignment algorithm [13]. Thus, in this work, we develop an assignment scheme using genetic algorithm to increase the reliability of a running system. The proposed scheme calculates a backup switch assignment map considering each controller might fail and taking load-balancing into account to adapt to new conditions if a failure occurs despite all efforts. It operates in an online manner and proactively determines assignment maps for different failure cases. It determines a backup controller for each forwarding node according to load of controllers, switch reassignment cost and maximum probability of failure for the shortest paths from a controller to its switches. In the case of controller failure, a switch will be assigned to its proactively calculated backup controller.

A. Assignment Parameters

For an assignment to support connectivity between network components and improve resilience, it should utilize existing network connectivity among switches. Moreover, the load distribution should be considered to minimize load-induced catastrophic incidents. The cost of assignment is also another factor since reassignment requires on-the-fly configuration of controller framework as well as switches. Therefore, to develop efficient reassignment algorithms with resilience objectives, probability of connectivity failure, controller load distribution and reassignment costs are important elements to be considered.

1) Probability of Connectivity Failure: If the connection between controller and the forwarding plane is broken because of network failures, some switches will be left without any controller and become stale for new flows. Network availability should be ensured to increase reliability of SDN. To formalize reliability for control plane availability, it can be defined as the probability of connectivity failure of a path from a switch to its controller as in (1):

$$\tilde{P}_n = 1 - \prod_{e_i, v_j \in R_{c \to n}} (1 - P_{e_i})(1 - P_{v_j}) \tag{1}$$

where P_{e_i} and P_{v_j} are probability of link failure and switch failures at a given time and $R_{c \to n}$ is the shortest path controller c to switch n.

2) Controller Load Distribution: Optimal load distribution directly impacts network performance, thus improving load balancing is important for network resilience [14]. In case of a controller failure, when the switches are assigned to other controllers, an ignorant assignment may lead some controllers to overload and even cause cascaded failures. Although there are various metrics for controller load such as CPU load or message queue lengths, we represent it as the total number of `PACKET_IN` messages (pim) to a controller from its switches. Then the controller load L_c is defined as $L_c = \sum_{i \in n} x_{c,i} R_n$ where the relevant assignment criterion is the load variance σ^2 calculated as $\sigma^2 = \frac{1}{c} \sum_{i \in c} (\overline{L} - L_i)$.

3) Reassignment Cost: When a controller fails, its switches will be distributed among other controllers. Reassignment cost $\tilde{R}$ is considered during calculation of reassignment map to measure the cost of applying that reassignment. It represents the trade-off between flexibility for assignments and the overhead of implementing them in a practical software-defined network. It can be calculated as in (2) as the total number of switches whose controllers are changed in case of failure when reassignment is performed. Specifically, x_{ij} is the previous assignment map and z_{ik} is the new assignment of switches after failure. We take XOR of two values, i.e. if a switch is assigned to a new controller $z_{ik} = 1$, otherwise $z_{ik} = 0$.

$$\tilde{R} = \sum_{i \in S, j \in C, k \in \tilde{C}} x_{ij} \oplus z_{ik} \tag{2}$$

The number of reassigned switches will be at least equal to the number of switches of the failed controller. Besides, when network size increases, it is likely that reassignment cost will also increase. Thus, when setting reassignment cost as a constraint, we limit it to be smaller than the total number of switches s multiplied with a scale factor γ.

B. PREF-CP Implementation

PREF-CP works proactively to enable the system to continue working efficiently in case of a controller failure in an SDN architecture. PREF-CP evaluates the current switch-to-controller assignment and calculates a backup map assuming that any controller might fail. For k controllers, k backup maps are calculated. When a reassignment is performed, PREF-CP recalculates the switch-to-controller assignment considering possible future failures and overload issues. It contains two main modules as depicted in Fig. 1 and explained below:

- **Monitoring Module (MM)** tracks controllers' health and pulls relevant statistics which contain current assignment

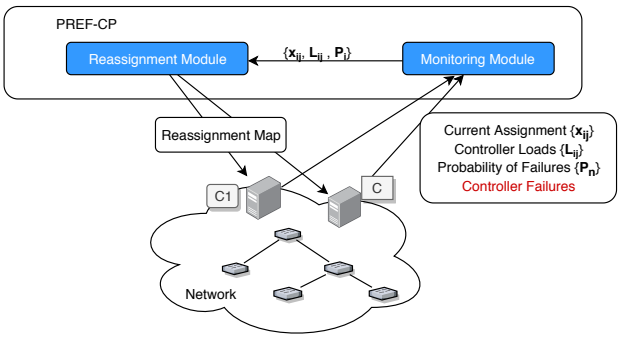

Fig. 1: PREF-CP architecture.

TABLE I: Model Parameters.

Symbol	Definition
P_v	Probability of switch failure
P_e	Probability of link failure
$\tilde{P}$	Probability of failure
$\tilde{R}$	Reassignment cost
α	Objective function weight coefficient
γ	Reassignment cost scale factor
s	Number of switches
σ	Standard deviation of controller load (PACKET_IN messages)
U_j	Maximum number of requests that controller C_j can handle
R_i	Number of requests of each switch i

x_{ij}, controller loads L_{ij}, and probability of connectivity failure values P_j. MM also detects failures in the control plane then informs PREF-CP reassignment module.

- **Reassignment Module (RM)** periodically checks the collected statistics from the monitoring module and calculates reassignment map accordingly. In case of controller failure, reassignment module performs new assignment based on that pro-actively determined backup map.

IV. Problem Formulation

A. System Model

In our system model, the SDN physical network is presented as a graph which is denoted as $G(V, E)$, where V is the set of nodes (switches and controllers) and E is the set of links. The set of controllers is denoted as $C \in V$, The subset of working controllers is denoted as $\tilde{C} \in V$ while the set of switches is denoted as $S \in V$. The model parameters are listed in Table I.

Objective. The goal of the proposed strategy is to jointly improve the controller load distribution uniformity and decrease probability of failure incorporating a weight coefficient to calibrate the priority of these elements:

$$\min(\alpha * \sigma + (1 - \alpha) * \tilde{P}) \tag{3}$$

Constraints.

A switch will be controlled by exactly one controller:

$$\sum_{j \in C} x_{i,j} = 1, \quad \forall_i \in S \tag{4}$$

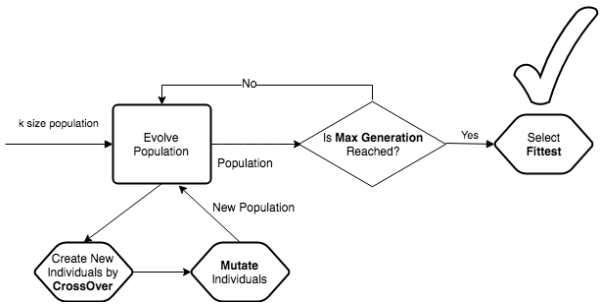

Fig. 2: Genetic algorithm.

Controller capacity cannot be exceeded:

$$\sum_{i \in S, j \in C} x_{i,j} R_i \leq (1 - \alpha) U_j \tag{5}$$

α values fall in range [0,1]:

$$0 \leq \alpha \leq 1 \tag{6}$$

The reassignment cost $\tilde{R}$ can not exceed the value calculated by the number of switches s multiplied by γ:

$$\sum_{i \in S, j \in C, k \in \tilde{C}} x_{ij} \oplus z_{ik} \leq \gamma \times s \tag{7}$$

V. Assignment Algorithms

A. Random Assignment

Although this is a trivial algorithm, it is typically used as a baseline case for performance evaluation. Moreover, it can be practical for situations where very low-complexity requirements are in force. In random assignment, each candidate controller has a uniform probability of hosting a switch. In that case, reassignment algorithm randomly chooses a controller among all potential ones (i.e., ones that are still running).

B. Genetic Algorithm Based Heuristic

A genetic algorithm is a search heuristic that reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation as shown in Fig. 2 [15].

1) Fitness Function: The performance of a genetic algorithm relies on how well a fitness function is derived. Fitness function in our approach considers minimization of maximum probability of failure of the shortest path of a switch to the controller ($\tilde{P}$) and the load distribution variance among controllers (σ). The fitness value $Eval = \alpha * \sigma + (1 - \alpha) * \tilde{P}$ is used during crossover for choosing which controller a switch will be assigned to for chromosome evaluation. That is, when the crossover operation has to choose a better gene from two parent genes during the crossover, it calculates fitness values of these genes in `ComputeFitness()` of Algorithm 1.

Algorithm 1 Genetic Algorithm

Require: Input: Topology, Load, CrossoverFunction, PopulationSize
Ensure: Generate solution x^c initialize R_{max} and S_{best}
1: Population ← InitializePopulation(PopulationSize)
2: Evaluate population
3: S_{best} ← BestSolution(population)
4: **while** population has not converged **do**
5: Selection : Parents ← SelectParent(Population, PopulationSize)
6: Children ← 0
7: Crossover()
8: ComputeFitness()
9: **if** Better fit **then**
10: PickParent()
11: **end if**
12: **end while**

C. Inter-Controller Greedy Algorithm (ICA)

This algorithm generates a list of potential assignments based on switch loads and failure probabilities of the shortest path from a switch to possible controller. It essentially sorts switches in an increasing order of their scores based on their failure probability and load values. Then, it chooses one switch at a time for assigning to a controller. The algorithm iterates until all switches are assigned as shown in Algorithm 2 [16].

Algorithm 2 ICA.

Require: Input: Topology G, Switch Load S, Number of Controllers N, Probability of Failures P
for i in Devices Size **do** score[i] = CalculateDeviceScore (L, P)
end for
Sort switches s in ascending order of their scores considering failure properties P_f and loads L_n as set S′
while Devices left to add to Assignment Map **do**
for j in Controllers Size **do**
Among all devices select the device lowest score from S′
Add switch s_i to Controller c_j's backup map
end for
end while
return AssignmentMap

VI. Performance Evaluation

We use ONOS Nightingale 1.13.1 as SDN controller and Mininet 2.2.1 for creating network topologies to implement the data plane. All simulations are run on a physical machine with Ubuntu 14.1 OS, 8GB RAM and Intel Core i7 6700HQ CPU. In our system, controllers are running inside Docker containers. We run Mininet on that physical machine and five ONOS instances in Docker 17.09 containers for simulating the distributed control plane. The switches run in Open vSwitch 2.0.2 mode to deliver OpenFlow functionality.

For network traffic, we use D-ITG traffic generator 2.8.1 and generate the traffic flows of VoIP, video and two game traffic types with Counter Strike characteristics related to the active phase of the game or an idle player [17]. We run our experiments on tree topologies T1:40 (i.e., tree topology with 40 switches), T2:85, T3:63, T4:121 and Internet2 OS3E topology T5 with 34 switches. T2 and T4 are specifically used for testing run-time performance of algorithms while T1, T3 and T5 are used for other experiments.

We run PREF-CP, ICA and RANDOM algorithms ten times for different α values in Table II, randomly failing one controller on each run and then measuring σ and $\tilde{P}$ values. We perform experiments on T1, T3 and T5 topologies and report mean values of σ and $\tilde{P}$ for each algorithm over these runs.

TABLE II: Simulation Parameters.

Parameter	Symbol	Value
Number of switches	s	{34, 40 , 63, 85, 121}
Weight parameter	α	{0.0, 0.5, 0.7, 1.0}
Maximum population size	θ	50
Number of controllers	N	5
Reassignment cost multiplier	γ	0.8

A. Simulation Results

1) Load Distribution: Load distribution characteristics can be affected by several network parameters; specifically reassignment algorithm, α value and the number of switches in the network. To see the effects of these factors, we compared the load distribution variance based on the number of PACKET_IN messages (pim) that are distributed among controllers. Increasing α increases the weight of the load distribution as defined in (3), thus load distribution improves (σ decreases) as α increases. The test results are shown in Fig. 3. PREF-CP distributes controller load by 8.38%, 11.5%, 16.6% better from ICA for α values 0, 0.5, and 1, respectively. Random assignment is plainly for reference and it is not affected by α since it assigns switches to random controllers without considering load distribution. For RANDOM algorithm, σ value equals to 9284.9 pim, which is higher than both PREF-CP with 3933.9 pim and ICA with 4438.9 pim. Although ICA algorithm outperforms random assignment, it is not as good as PREF-CP.

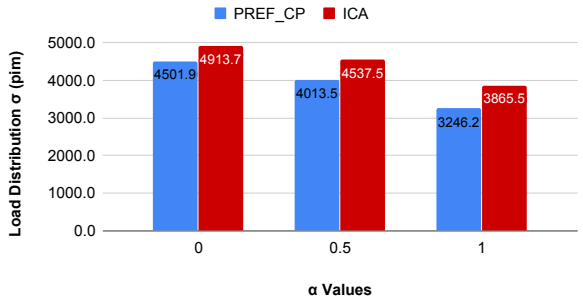

Fig. 3: Load distribution variance σ for different algorithms.

The results of normalized values for load distribution of PREF-CP on different sized topologies T1:40, T3:63 and T5:34 are listed in Table III. For $\alpha = 0$, topology sizes affect load distribution performance more than for $\alpha \geq 0.5$ case. Increasing the number of switches does not negatively impact the performance of PREF-CP algorithm. For greater α values, the network size is less important at load distribution performance of PREF-CP algorithm.

2) Impact of α on PREF-CP performance: As we change α to prioritize load distribution or probability of failure in our optimization objective, these components behave as expected. When α is 0, PREF-CP tries to find an assignment algorithm

TABLE III: PREF-CP Load Distribution for varying s.

Topology	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
T5-34 switches	0.373	0.360	0.356
T1-40 switches	0.369	0.363	0.356
T3-63 switches	0.364	0.359	0.350

TABLE IV: PREF-CP Performance - Load Distribution σ (pim) and Probability of Failure $\tilde{P}$.

Objective Function	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
Load Distribution σ	0.371	0.359	0.356
PoF $\tilde{P}$	0.389	0.433	0.465

with lower PoF. As stated in constraint (6), α values range in [0,1]. As seen in Table IV, as α increases, PREF-CP prefers an assignment map favoring load distribution σ over $\tilde{P}$.

3) PoF ($\tilde{P}$) Characteristics: We compare the probability of connectivity failure ($\tilde{P}$) for different algorithms to investigate their performance. There are basically two aspects in a network that affect PoF. First one is α; lower α values result in better PoF. As seen in Fig. 4, PREF-CP outperforms ICA and random assignment algorithms. ICA algorithm performs slightly better than random counterpart which has PoF 0.577. However, the random algorithm is for reference since it assigns switches randomly regardless of load or PoF.

The second one is, as seen in Fig. 5, the number of hops in the test topology: when that value increases, probability of failure is also increasing. This outcome is expected because PoF directly depends on the number of switches and links between from *node a* to *b*. If we compare two tree topologies T1 and T3, T1 has depth of 4 and T3 has depth 6, and average PoF values are 0.373 and 0.509, respectively, when their PoF values are averaged over different α values in the figure. As α increases, the weight of PoF decreases which results in higher $\tilde{P}$ values: average PoF values of topologies T1, T3, and T5 are 0.389, 0.433, and 0.465 with increasing α values.

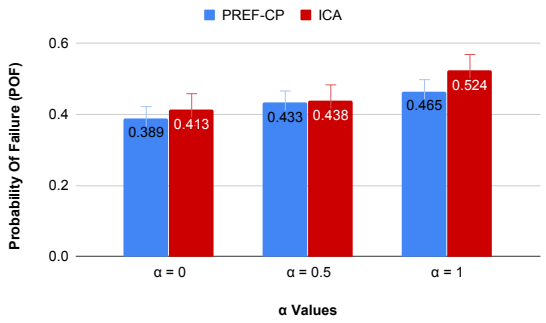

Fig. 4: $\tilde{P}$ values calculated by different algorithms.

4) Computational Time: For the complexity aspect, computational time of the proposed PREF-CP algorithm was compared with ICA for different network sizes, i.e. number of switches s, to see the effect. As seen in Table V, when s increases in the network, calculation time for reassignment map also increases. For PREF-CP, it is substantially higher

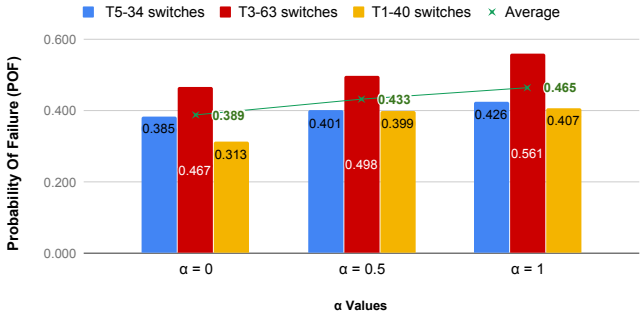

Fig. 5: $\tilde{P}$ values for different topologies.

TABLE V: Run-times for Different Assignment Algorithms (in msec).

Assignment Algorithms	Number of switches		
	40	85	121
GA	2480	7540	12100
ICA	12	27	80

compared to ICA algorithm as expected. This result renders the complexity-assignment quality trade-off since PREF-CP is most time-consuming albeit being better in optimization. However, please note that we have not applied any hardware/software optimizations to accelerate GA computations.

B. Cascaded Failure Characteristics

When two controllers fail in tandem, the residual load is distributed among the remaining $V - 2$ (three in our case) controllers. The presented results shown in Fig 6 are the average of load distributions with respect to topologies. Tree topologies (T1, T3) are affected by multiple controller failures more than Internet2 OS3E topology (T5). Fig. 7 shows load distribution for all algorithms in case of one controller failure and two controller failures. PREF-CP algorithm outperforms both random and ICA algorithms.

To see more detailed how control load is distributed on each individual controller, we examine load distribution of `PACKET_IN` messages on controllers in Internet2 OS3E topology with 34 switches for $\alpha = 0.5$. In this case, C3* and then C1* fails one after another and we examine PREF-CP performance in handling two controller failures. For *before-*

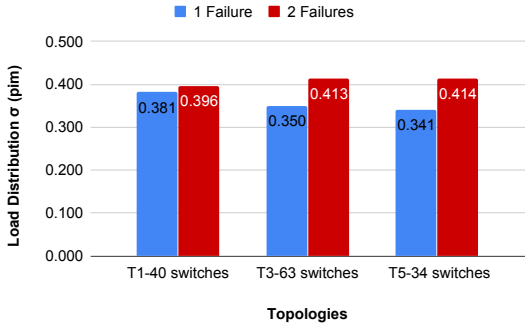

Fig. 6: Load distribution under controller failures for different topologies.

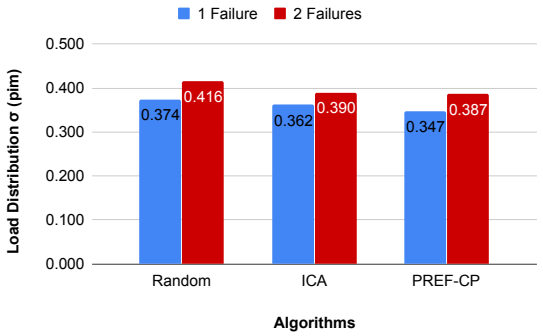

Fig. 7: Load distribution under two controller failures for different algorithms.

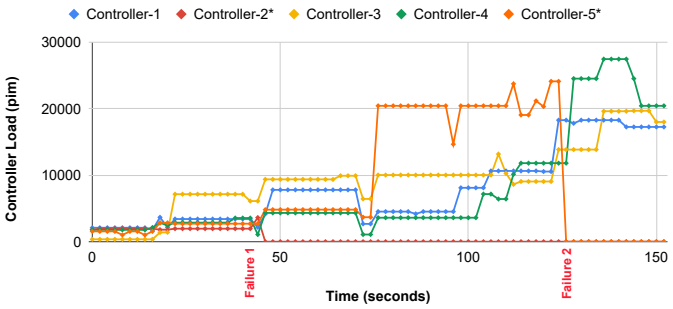

Fig. 8: Controller load vs time - PREF-CP ($\alpha = 0.7$, $\gamma = 0.8$).

failure situation, the load distribution σ is 4260.1 pim. When the first failure occurs (*after-one-failure* regime), the load variance decreases to 3773.9 pim while it again increases after the second failure to a higher value of 5182.2 pim (*after-two-failures* regime). Therefore, PREF-CP is quite successful to maintain the load distribution uniformity level although the load per controller increases due to fewer operating controllers.

C. Controller Load Behavior Over Time

To see the effect of controller failures and consecutive reassignment, we observe the loads of individual controllers over the experiment period. This graph is an output of PREF-CP run with parameters $\alpha = 0.7$ and $\gamma = 0.8$ on topology T5-34 switches. Fig. 8 shows messaging load changes of controllers in time. C2 fails at time $t = 40$ sec, and then its switches are distributed to other controllers. Control traffic loads of remaining controllers surge after the reassignment as expected. At $t = 125$ sec, second failure occurs where the controller with the highest load (C5) is taken offline. Again the loads for remaining controllers increase overall.

VII. Conclusion

In this paper, we consider control plane failures and how to recover from them with an efficient proactive approach. To address this challenge, we propose a proactive switch assignment scheme PREF-CP in case of controller failures using genetic algorithm considering controller load distribution, reassignment cost and probability of failure. Our test results show that when controllers' load distribution and probability of connectivity failure are considered, PREF-CP performs better than alternative ICA algorithm. For future work, experiments can be run on different topology types and larger network sizes. For a more comprehensive comparison, latency and throughput results can also be considered.

Acknowledgment

This work was supported by the Scientific and Technical Research Council of Turkey (TUBITAK) under grant number 117E165.

References

[1] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[2] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, July 2013.

[3] M. Özçelik, N. Chalabianloo, and G. Gür, "Software-defined edge defense against IoT-based DDoS," in *2017 IEEE International Conference on Computer and Information Technology*, Aug 2017, pp. 308–313.

[4] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 7–23, Apr. 1999.

[5] M. Bari, A. Roy, S. Chowdhury, Q. Zhang, M. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *2013 9th International Conference on Network and Service Management (CNSM)*, Oct 2013, pp. 18–25.

[6] M. Obadia, M. Bouet, J. Leguay, K. Phemius, and L. Iannone, "Failover mechanisms for distributed SDN controllers," in *2014 International Conference and Workshop on the Network of the Future (NOF)*, vol. Workshop, Dec 2014, pp. 1–6.

[7] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. ACM, 2012, pp. 7–12.

[8] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: An enhanced controller placement strategy for improving SDN survivability," in *2014 IEEE Global Communications Conference*, Dec 2014, pp. 1909–1915.

[9] K. Fang, K. Wang, and J. Wang, "A fast and load-aware controller failover mechanism for software-defined networks," in *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, July 2016, pp. 1–6.

[10] K. Nguyen, Q. T. Minh, and S. Yamada, "A software-defined networking approach for disaster-resilient WANs," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, July 2013, pp. 1–5.

[11] N. Beheshti and Y. Zhang, "Fast failover for control traffic in software-defined networks," in *2012 IEEE Global Communications Conference (GLOBECOM)*, Dec 2012, pp. 2665–2670.

[12] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, Sep. 2016, pp. 253–259.

[13] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On the placement of controllers in software-defined networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, Supplement 2, pp. 92 – 171, 2012.

[14] Y. Zhou, Y. Wang, J. Yu, J. Ba, and S. Zhang, "Load balancing for multiple controllers in SDN based on switches group," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Sept 2017, pp. 227–230.

[15] M. Gen and R. Cheng, *Genetic Algorithms*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1999.

[16] S. Güner, H. Selvi, G. Gür, and F. Alagöz, "Controller placement in software-defined mobile networks," in *23nd Signal Processing and Communications Applications Conference (SIU)*, 2015, pp. 2619–2622.

[17] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.