Computer Networks 254 (2024) 110819

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

An innovative NSGA-II-based Byzantine Fault Tolerant solution for software defined network environments

Waqas Ahmed [a], Nadir Shah [a,*], Gabriel-Miro Muntean [b]

[a] Department of Computer Science, COMSATS University Islamabad, Wah Campus, Pakistan
[b] School of Electronic Engineering, Dublin City University (DCU), Glasnevin Campus, Dublin 9, Ireland

ARTICLE INFO

Keywords:
Byzantine Fault Tolerance (BFT)
NSGA-II
Link reliability

ABSTRACT

Byzantine fault tolerance (BFT) of the control plane in Software Defined Networking (SDN) is achieved by mapping each switch to $3f + 1$ number of controllers, where f represents the number of faulty controllers that can be tolerated at a time. A BFT approach protects the data plane from any potential malicious activity at the control plane by detecting the inconsistency among the response messages from multiple controllers. To compute the optimal mapping of switches to the controller, the existing literature does not consider some important parameters. This paper proposes a novel approach, named NBFT-SDN, that extends an artificial intelligence algorithm (*i.e.* NSGA-II) to solve a new formulated multi-objective optimization problem associated with this mapping. NBFT-SDN considers the very important parameters link reliability and link load along with switch-to-controller minimum delay, switch-to-controllers maximum reliability, controller-to-controller minimum delay, minimum link load, minimum hop count, and controller load balancing when mapping the switches to the controllers in optimum manner. The performance of our proposed approach is evaluated in comparison to a state-of-art approach using real network traces with network topologies of diverse sizes. Our proposed approach NBFT-SDN show improved network performance in terms of reliability, delay, hop count, load balancing and link load.

1. Introduction

Software Defined Networking (SDN) enables easy network control and management by decoupling the control plane from data plane and implementing the control plane for all the devices at a logically central entity, called the SDN controller. The data plane consists of SDN switches that forward data as per the instructions of the SDN controller. The SDN controller is therefore responsible for the entire data plane. It allows for an overall view of the entire network and enables centralized configuration and easy policy enforcement (*i.e.* centered at the controller) [1]. This SDN programmable nature allows easier integration of SDN with other emerging technologies such as Network Function Virtualization (NFV), Internet of Things (IoT) and Cloud Computing. Despite of its obvious advantages, SDN is vulnerable to Single Point of Failure (SPoF) of the controller due to software bugs, miss-configuration or increased data plane requests that overwhelm single controller to respond quickly. To address these issues, a distributed control plane was proposed, where more than one controller manages the data plane. The resulting SDN multi-controller architecture solves the scalability problem of SDN and provides fault resilience. However, there are several research challenges including in relation to controller inter-connectivity and placement to optimize the network performance, mostly in terms of delay, but also in terms of fault tolerance [2–4]. Multi-controller architectures often use open source controllers such as OpenDayLight and ONOS and they rely on consensus algorithm such as RAFT [5] to address state synchronization. RAFT is vulnerable to malicious controller attacks, that can inject malicious messages in the network. These messages may route the traffic to paths different then the ones computed by legitimate controllers. The RAFT algorithm is based on a single leader design that can distribute configuration updates to the followers and therefore a malicious leader may inject false configuration updates and even block correct replies from followers. This behavior is known as a byzantine failure of nodes and such an attack can compromise the entire network.

To address this issue (*i.e.*, and make the network perform correctly even in the case of a controller becoming compromised), byzantine agreement-based solutions, called byzantine fault tolerant (BFT) approaches are proposed [6–9]. In a BFT approach, a switch is mapped to a certain number of controllers (*i.e.* k). The switch exchanges the state information with all k controllers. Similarly, a switch sends a flow setup request to all k controllers. When the switch receives the flow

* Corresponding author.
E-mail address: nadirshah82@gmail.com (N. Shah).

https://doi.org/10.1016/j.comnet.2024.110819
Received 13 June 2024; Received in revised form 18 September 2024; Accepted 18 September 2024
Available online 23 September 2024

rules (*i.e.* route information) against a flow setup request from the k controllers, the switch compares the flow rules. There are two possible cases: (i) If the switch finds that the flow rules of k controllers are consistent, then it means none of its k controllers are compromised, and subsequently installs the flow rules and performs the corresponding action. (ii) Otherwise, when the switch receives different flow rules from the k controllers, it identifies the compromised controller based on a majority rule: the majority of controllers which have the same flow rules, are considered not compromised, whereas the remaining ones, are. The switch advertises the compromised controller(s) so that all the switches assigned to the compromised controller will be disassociated from it and be associated to existing non-compromised controllers and/or new controllers. Some existing BFT approaches [6,9,10] map $3f + 1$ controllers per switch to tolerate f malicious controllers, while other approaches [7,8] map $f + 1$ primary and f secondary controllers per switch. These are good solutions, but overlook important performance-related aspects like the link reliability and congestion delay in their switch-controller mapping process.

In SDN, link reliability is a critical concern because of path adjustments made by dynamic control plane which are vulnerable to disruptions [11]. Work in [12] shows that link failure frequency is every 30 min and these failure can lead to cascading disruptions, that cause packet loss and increased recovery time [13]. Link failure has a detrimental impact on applications, especially those involving latency-sensitive and real-time communications [12,13], and the BFT. Existing BFT approaches [7–9] overlook the link failure while mapping data plane switches to controllers. Links are important as they carry data packets and control packets such as synchronization messages between controllers in in-band configuration. Moreover, a single link can be a medium to transport synchronization messages from different $3f + 1$ controller clusters. In such a situation, obtaining a reliable switch-to-controller mapping has paramount importance because link failure causes loss of data and control packets and can lead to synchronization process failure between controller clusters.

To fill the gap of the existing research, **we propose a multi-objective evolutionary computational approach based on the Non-Dominated Sorting Genetic Algorithm (NSGA-II) for performance-oriented switch-controller mapping to tolerate byzantine faults**. The proposed approach aims at maximizing the data plane reliability, minimizing the path delay for switch-to-controller and controller-to-controller communications, minimizing hop count on the path between switch-to-controller, minimizing the link load, and maximizing the load balancing over the controllers. NSGA-II [14] is a multi-objective optimization algorithm, which ensures the diversity in the population in order to explore the search solution space efficiently. To the best of the authors' knowledge, we are the first to employ the NSGA-II algorithm for controller placement in the context of BFT in SDN, especially considering link reliability and link load which are totally overlooked by the existing approaches. The proposed solution was named NSGA-II-based Byzantine Fault Tolerant approach for SDN (NBFT-SDN) and the paper's contributions are described as follows.

- This paper considers the link reliability and link load for controllers placement in the context of BFT in SDN. To the best of our knowledge these parameters are not used by other existing approaches.
- We formulate controller mapping as a conflicting multi-objective problem. Mapping solutions obtained via heuristic based approaches may not be optimal or using solutions based on linear programming may not be suitable. To the best of our knowledge we are the first to use the NSGA-II algorithm to address the BFT mapping problem in SDN.
- We modify the random selection process for initial population generation by suggesting a new approach to generate random switch-controller mapping population that satisfies the unique requirement of mapping a switch to $3f+1$ controllers. This enable us to reduce the computation time of the algorithm.
- Additionally, we also modify the cross-over and mutation operators of NSGA-II in our target scenario to prevent solution becomes infeasible after crossover between two parents and mutation of offspring solution.
- We include in our algorithm a good percentage of both infeasible and feasible solutions in the search space, to prevent the algorithm to find the best solution from the feasible search space only.
- We evaluate the performance of NBFT-SDN on a wide range of real small-to-large publicly available typologies [15,16] and datasets [17], which was not done by existing approaches.

The rest of paper is organized as follows: Section 2 discusses related works, Section 3 describes the mathematical modeling of the proposed approach NBFT-SDN, Section 4 presents NBFT-SDN in details and Section 5 includes the evaluation in comparison with the existing approaches. In Section 6, we present some challenging scenarios related to our proposed solution. Finally, Section 7 concludes the paper.

2. Related work

In this section, we give an overview of some existing related works in order to clearly highlight their limitations and the significance of our contribution. We divided the existing approaches into different categories as follows.

2.1. $(3f + 1)$ Controllers

In this section, we describe the BFT solutions where a switch is mapped to $3f + 1$ number of controllers.

The BFT solution proposed in [18] reduces the computational cost of message comparison by offloading this process to programmable switches. That is these programmable switches compare the configuration response sent by the controllers and commit to data plane if $f + 1$ identical Messages received. Moreover, the Message ordering in it is ensured via appending time stamp with each packet. It does not consider controller state synchronization before committing data plane update and does not consider the link reliability and the link load while mapping the switches to controllers.

To extend the solution proposed in [18], the BFT solution in [19] targets a scenario where some of the controller replicas are slower than others. Hence it groups the controllers into several groups depending upon their processing speed. Then it achieves consensus among the controllers in different groups to increase throughput of the overall network in the context of BFT. However, this solution does not take path reliability in mapping a switch to controller and thus does not avoid/reduce loss of synchronization messages due to link reliability.

The BFT solution in [6] introduces a proxy layer between distributed control plane and data plane. The proxy layer collects and compares the $f+1$ number of identical messages from multiple controllers. If the messages are consistent, configuration message is committed on target switch. Otherwise, the faulty controller is identified. This approach does not quantify the scalability of proxy layer in-terms of important mapping parameters such as link delay, link reliability and link load between data plane and control plane.

In the BFT solution proposed in [9], each switch is mapped to $2F_M + F_A + 1$ controllers where (F_A) represent the controller failures related to their availability, and (F_M) represents the byzantine failure (F_M). Further, to obtain switch-to-controller mapping using integer linear programming (ILP) [20], this solution considers the delay between switch and controller, and controller processing capacity as objective functions. However, this solution does not take link reliability or link load parameters to find the optimal mapping for switch-to-controller.

The BFT solution proposed in [21] proposes a SDN-tree to hold message exchange during BFT consensus to deduce faulty and corrupted controllers. However, this solution does not emphasize on parameters like link reliability, link load and load-balancing for required for switch-controller mapping.

The BFT solution proposed in [10] considers the switch level granularity for byzantine fault tolerance, *i.e.* every switch decides its number of faulty controllers (f) to be tolerated by the switch independent from other switches demand for the BFT. However, this approach does not consider the link reliability, link load and load balancing among the controllers in mapping the switch to controllers for BFT.

2.2. $(2f+1)$ Controllers

In this section, we describe the BFT solutions which map a switch to $2f+1$ number of controllers.

The BFT solutions proposed in [7,8] map a switch to $f+1$ number of primary and f number of secondary controllers such that minimum number of controllers are used, the switch-to-controller delay is minimized and the load among the controllers is balanced. Their proposed approach works as follows. First, a packet-in (a request for flow rules for a data flow) message is sent to $f+1$ primary controllers. The response from the secondary controllers is only required if any inconsistency is found in the response from primary controllers. This BFT solution generates low traffic load, however, the controllers states are not synchronized. Additionally, it overlooks the link reliability and the link load in computing the placement for controllers.

2.3. Non-BFT approaches

As we have considered in our proposed approach the reliability of controller-to-controller path in mapping a switch to controllers. This is because the controllers exchange the messages with each other for achieving the consensus among their status. Therefore, in this section, we describe the existing approaches for the reliability and the consensus among the controllers in multi-controller environment so that we can differentiate the significance of our proposed approach. These approaches do not consider the BFT for SDN controllers.

The study in [22] achieves consensus among the controllers via the PAXOS [23] algorithm. The consensus procedure is carried in virtual network function (VNF) based to reduce CPU overhead. Each replica controller offloads its consensus procedure to VNF. Though this study suggests an approach to achieve the consensus in an efficient way among the controllers in a multi-controller SDN context, however, it does not apply in the BFT scenario which has unique requirements.

The authors of [24] propose a solution to compute reliable controller in multi-domain switch-controller arrangement. This solution relies on heart beat messages to detect failure of particular controllers. In case a controller is detected as faulty, the switches assigned to the failed controller will be managed by another controller with high reliability. By reliability of the controller, the authors mean the number of times a controller fails. They do not consider the reliability of the path connecting the switches to their controllers. The study in [25] proposes a gossip protocol to detect controller failure that relies on heartbeat messages between nodes. A failure detector estimates the inter-arrival time of next expected heartbeat and decides the controller failure if half of the nodes in the cluster agree with the failure of a particular node. However, the proposed solution does not detect the byzantine failure of controllers.

Ibrar et al. [17] targets a hybrid SDN, which is a network where SDN switches are deployed along with legacy switches in the network. The proposed approach predicts link reliability using a K-NN algorithm [26] and historical link failure values. The historical values of every link are represented as a 5-tuple (*daytime, linkDowntime, linkUptime, linkDownfrequency*). The reliability value of each link is computed in each time slot and predicted for a new link-down event. Further, the authors compute the path from source to destination using a deep reinforcement learning algorithm [27] based on several parameters, such as link bandwidth utilization, link delay, and reliability of the link. The same authors have extended their previous work [17], and in [28] they have computed Pareto-optimal paths between IoT devices and Fog servers using the NSGA-II algorithm.

To summarize the research gaps, a comparison is provided in Table 1 to emphasize the selection of different parameters in our proposed approach NBFT-SDN, Next are the limitations of the existing approaches.

- First, some important parameters like the reliability and utilization of paths (a path consists of a number of links) connecting switches to their corresponding controllers are not considered. We aim to generate robust controller mapping against diverse network events [30] so that packet loss is minimized and flow setup requests traverse less loaded links.
- Second, to compute the correct and efficient mapping of switches to controllers in a BFT context using many parameters, it is an optimization problem [31]. To address this, the existing approaches are mostly using general optimization algorithms like integer linear programming [7,9,19,29]. Such optimization algorithms are suitable for single objective optimization and they combine the multiple objectives into a single objective by using the techniques like normalization. However, this can miss the importance of some parameters. So these existing optimization algorithms are not suitable for multi-objective optimization problems. By considering the link reliability and link load along with other parameters (mentioned in Table 1 for computing an efficient mapping of the switches to the controller in the context of BFT) a multi-objective optimization approach, like NSGA-II [14] is the best solution.
- Third, our target problem involves several objectives as described in Section 3.2 and constraints 4.3 which are conflicting in nature. We opt for an AI-based multi-objective approach, like NSGA-II [14], to simultaneously optimize all the objectives.
- Fourth, though some of the existing approaches are evaluated on real network typologies [9], however, they do not consider the real traffic data of these typologies. So the results of their proposed approaches cannot be well judged in the absence of the real time traffic.

Next, the target problem is presented in the context of an example scenario, that represents computation of $3f+1$ controllers for a particular data plane switch to tolerate byzantine faults.

3. Byzantine Fault Tolerant (BFT) problem in SDN

As we have explained in Section 2, the existing approaches for BFT in SDN overlook important features like link reliability and link load for controller placement. In the following section, we show through an example scenario the importance of our problem statement.

3.1. Example scenario

First, let us assume a simple network topology consisting of five switches and six controllers as shown in Fig. 1. The switches are connected with each other using links. Each link is labeled with different weights defining QoS constraints such as link reliability $r_{e(k,l)}$ and transmission delay $t_{e(k,l)}$. A switch i can reach any controller via path(s) having different link reliability and delay values. For instance, the source switch $sw(i)$ can reach controller C_2 and C_5 via three different paths shown in Table 2. Assume each switch needs to map to $3f+1$ controllers for BFT operation, and $f = 1$, as given in [8,9]. Our objective is to map the switches to controllers such that the path from a switch to its corresponding controllers have minimum transmission delay, maximum reliability and minimum hop count. All of these objectives are conflicting in nature and we need to find optimal $3f+1$ controllers for each switch in the data plane. Table 2 shows parameter

Table 1
Existing literature comparison with proposed approach (NBFT-SDN)

Property	[8]	[7]	[9]	[29]	[6]	[19]	[24]	[21]	NBFT-SDN
Switch-Controller Assignment	✓	✓	✓	✓	✓	✓	✓	×	✓
In-band Control Plane	✓	✓	✓	✓	✓	✓	×	×	✓
Controller Capacity	✓	✓	✓	×	×	✓	×	×	✓
Path Delay	✓	✓	✓	✓	×	✓	✓	×	✓
Path Reliability	×	×	×	×	×	×	✓	×	✓
Controller Synchronization	×	×	✓	✓	✓	✓	×	✓	✓
BFT (Byzantine Fault Tolerance)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Load Balancing	✓	✓	×	×	×	×	×	×	✓
Link Load	×	×	×	×	×	×	×	×	✓
AI (Artificial Intelligence)	×	×	×	×	×	×	×	×	✓

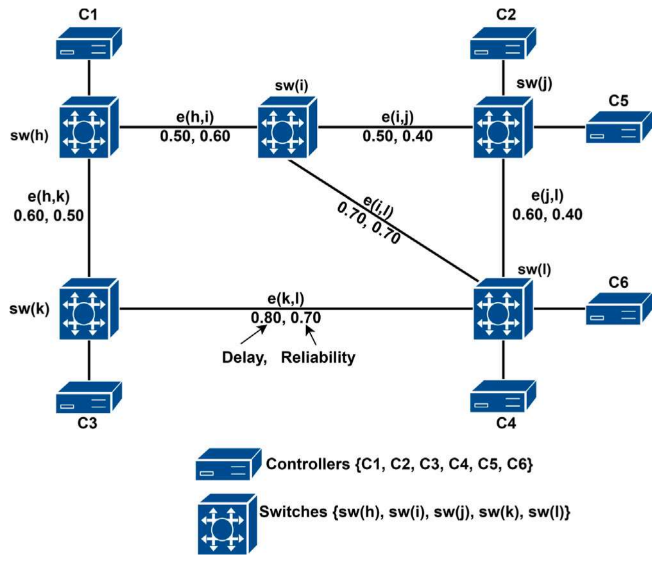


Fig. 1. A multi-controller example topology.

Table 2
Parameter computation.

Switch i to C_1 Paths	$R_{P(s_h,c_d)}$	$T_{P(s_h,c_d)}$	$HC_{P(s_h,c_d)}$
$p_1,(s_i,c_1)=\{s_i,s_h,c_1\}$ $p_2,(s_i,c_1)=\{s_i,s_l,s_k,s_h,c_1\}$ $p_3,(s_i,c_1)=\{s_i,s_j,s_l,s_k,s_h,c_1\}$	$max(0.60,0.50,0.40)$ $=0.60$	$min(0.50,2.0,2.5)$ $=0.50$	$min(1,3,4)$ $=1$
Switch i to C_2 Paths	–	–	–
$p_1,(s_i,c_2)=\{s_i,s_j,c_2\}$ $p_2,(s_i,c_2)=\{s_i,s_l,s_j,c_2\}$ $p_3,(s_i,c_2)=\{s_i,s_h,s_k,s_l,s_j,c_2\}$	$max(0.40,0.40,0.40)$ $=0.40$	$min(0.50,1.3,2.5)$ $=0.50$	$min(1,2,4)$ $=1$
Switch i to C_3 Paths	–	–	–
$p_1,(s_i,c_3)=\{s_i,s_h,s_k,c_3\}$ $p_2,(s_i,c_3)=\{s_i,s_j,s_l,s_k,c_3\}$ $p_3,(s_i,c_3)=\{s_i,s_l,s_k,c_3\}$	$max(0.50,0.40,0.70)$ $=0.70$	$min(1.1,1.9,1.5)$ $=1.1$	$min(2,3,2)$ $=2$
Switch i to C_4 Paths	–	–	–
$p_1,(s_i,c_4)=\{s_i,s_l,c_4\}$ $p_2,(s_i,c_4)=\{s_i,s_j,s_l,c_4\}$ $p_3,(s_i,c_4)=\{s_i,s_h,s_k,s_l,c_4\}$	$max(0.70,0.40,0.50)$ $=0.70$	$min(0.70,1.1,1.9)$ $=0.70$	$min(1,2,3)$ $=1$
Switch i to C_5 Paths	–	–	–
$p_1,(s_i,c_2)=\{s_i,s_j,c_2\}$ $p_2,(s_i,c_2)=\{s_i,s_l,s_j,c_2\}$ $p_3,(s_i,c_2)=\{s_i,s_h,s_k,s_l,s_j,c_2\}$	$max(0.40,0.40,0.40)$ $=0.40$	$min(0.50,1.3,2.5)$ $=0.50$	$min(1,2,4)$ $=1$
Switch i to C_6 Paths	–	–	–
$p_1,(s_i,c_6)=\{s_i,s_l,c_6\}$ $p_2,(s_i,c_6)=\{s_i,s_j,s_l,c_6\}$ $p_3,(s_i,c_6)=\{s_i,s_h,s_k,s_l,c_6\}$	$max(0.70,0.40,0.50)$ $=0.70$	$min(0.70,1.1,1.9)$ $=0.70$	$min(1,2,3)$ $=1$

computation from switch i to all the controllers, thus our aim is to sort optimal 4 among 6 controllers to map switch i. We compute functions delay and reliability on all paths *i.e.* $\{p_1(s_k,c_5),p_2(s_k,c_5),p_3(s_k,c_5)\}$ and map the switches to the controllers such that each switch has the best path available (in terms of minimum transmission delay, maximum reliability and minimum hop count) to its corresponding controllers. We shall define these functions in the next section.

For switch i, using our proposed approach we computed 4 controllers $\{c_1,c_3,c_4,c_6\}$ in contrast to other existing solutions like [9] that rely on a single main objective *i.e.* delay between switches and would output a different set of controllers such as $\{c_1,c_3,c_2,c_5\}$. This is because, as we consider link reliability, in this case the path from $sw(i)$ to c_4 and c_6 is a much more reliable path than to c_2 and c_5. This selection of controllers will cause at least 30% increase in successful packet delivery ratio. Additionally, we also assume that all controllers can reach each other via the same control path as we consider in-band mode of communication. As our selected controllers synchronize their states using more reliable paths as compared to the existing approaches like [9], the packet loss due to the link failure is reduced.

As the problem suggests we have conflicting objectives to solve, such as switch-to-controller delay minimization, switch-to-controller reliability maximization, controller-to-controller delay minimization, and controller-to-controller reliability maximization. Our problem needs a robust multi-objective algorithm that explores a large search space in the minimum time. ϵ-MOEA can provide a uniformly distributed set of solutions in the search space. However, its convergence is sensitive to the choice of value for the ϵ parameter. If the value of ϵ is too large, the algorithm may converge into a sub-optimal region of the search space. If the value of ϵ is small, then it makes the algorithm computationally expensive. Similarly, the Particle Swarm Optimization (PSO) lacks a dedicated mechanism to preserve diversity in the solution space during exploration. That leads to sub-optimal convergence before reaching the optimal solution [32]. To address these issues, NSGA-II introduces diversity in the solution space by a non-dominated sorting and crowding distance mechanism. Non-dominated sorting is computationally faster, and sorts the individuals by their Pareto dominance. The crowding distance mechanism finds the solutions with less crowded regions (higher crowding distance) in the search space over the different generations. Even if solutions belong to the same non-dominating front, solutions are prioritized, *i.e.* the solutions with higher crowding distance are considered for the next generation of the population.

Similarly, other techniques like Integer linear programming (ILP) used in MORPH [7,9] and the heuristic algorithm used in Pri-Backup [8] used for switch-to-controller mapping in the context of BFT also have limitations, as follows. These techniques use only a small subset of the search space and return sub-optimal solutions [33]. ILP solves problems involving a single objective and cannot solve non-linear problems. However, our proposed problem NBFT-SDN involves conflicting objectives such as maximizing network reliability and minimizing path delay. Optimizing such conflicting objectives involves a high degree of dispersion of data [33]. An evolutionary algorithm, such as NSGA-II, finds the solutions dealing with such conflicting objectives and explores a large search space effectively to find near-optimal solutions that are distributed across the Pareto-optimal front.

NSGA-II involves elitism, a non-dominated sorting mechanism to preserve the best-performing solutions for the next generation. It evaluates solutions by all objectives and sorts them. This sorting results in solutions division in different optimal fronts that are better in at least one objective of the problem and not worse in any other objective. The proposed approach is formally described in the next Section 3.2, where we define the mathematical notations, Objective functions, and Problem constraints regarding our proposed approach NBFT-SDN.

3.2. Problem formulation

Consider we have a network topology consisting of switches and controllers represented by an undirected graph $G(V, E)$ where the set $V = \{v_1, v_2, \dots, v_n\}$ represents the set of switches and controllers. The set of switches is $S = \{s_1, s_2, \dots, s_m\}$, and the set of controllers is $C = \{c_1, c_2, \dots, c_o\}$ such that $V = S \cup C$. The set $E = \{e_1, e_2, \dots, e_p\}$ represents the edges connecting a switch to a switch or a switch to controller. That is an $e_i = (v_j, v_k)$ where either both v_j and $v_k \in S$ (*i.e.* $e_i = e(s_j, s_k)$), or $v_j \in C$ and $v_k \in S$ (*i.e.* $e_i = e(c_j, s_k)$), or $v_j \in S$ and $v_k \in C$ (*i.e.* $e_i = e(s_j, c_k)$). Both v_j and $v_k \notin C$.

Eq. (1) defines the path from a switch s_k to a controller c_d consisting of a set of edges that connect nodes (switches and the controller). This path consists of switches and the controller c_d which can also be represented as in Eq. (2).

$$p_{(s_h,c_d)} = \{e(s_h, s_i), e(s_i, s_j), \dots, e(s_k, s_d)\} \tag{1}$$

$$p_s, (s_h, c_d) = \{s_h, s_i, \dots, s_k, c_d\} \tag{2}$$

where s_h is the source switch and s_k is the destination switch to which the controller c_d is directly connected to. Each switch can reach a controller by multiple available paths. So a set of paths from s_h to c_d are represented as:

$$P_{(s_h,c_d)} = \{p_1(s_h, c_d), p_2(s_h, c_d), \dots, p_{q,(s_h,c_d)}\} \tag{3}$$

Eq. (3) formulates the finite q number of paths between switch s_h and controller c_d, where $p_{q(s_c,c_d)}$ represents the qth path. To find the maximum reliable paths in our proposed solution we followed [17] to label every link with its respective reliability value, calculated based on 5 distinct link features collected from the real network operation against each link. These features include (a) *daytime*: recorded event time, (b) *linkDowntime*: link failure starts time, (c) *linkUptime*: the time when the link recovered from failure, (d) *linkDownfequency*: frequency of link failure, (e) *linkFailureCause*: link failure cause. Reliability of the edge between switch s_h and s_i is represented as $r_{e(s_h,s_i)}$, whereas the set R_p includes the minimum reliability of all edges forming the path from s_h to c_d in Eq. (4).

Similarly, we define the set R_P with the maximum reliability of all q paths originating from s_h to c_d in Eq. (5), where $R_{p_q,(s_h,c_d)}$ represents the reliability of the qth path. We also compute a path with the maximum reliability among all q paths originating from s_h to c_d as in Eq. (6), where the set R^* represents the path from switch s_h to c_d with the maximum reliability.

$$R_{p(s_h,c_d)} = \min(\{r_{e(s_h,s_i)}, r_{e(s_i,s_j)}, \dots, r_{e(s_k,s_d)}\}) \tag{4}$$

$$R_{P(s_h,c_d)} = \max(\{R_{p_1(s_h,c_d)}, R_{p_2(s_h,c_d)}, \dots, R_{p_q,(s_h,c_d)}\}) \tag{5}$$

$$R^* = \max_{\forall c_d \in C} \{R_{P_{(s_h,c_d)}}\} \tag{6}$$

We model the transmission delay of a link as the fraction t of the time required to send a packet of length l_{pkt} on the transmission link with the available bandwidth b_{li}. The transmission delay of an edge between s_h and s_i is represented as $t_{e(s_h,s_i)}$ whereas the set T_p includes the transmission delay of a path between switch s_h and c_d as shown in Eq. (7). A path with the minimum transmission delay between s_h to c_d is represented in Eq. (8). Similarly, we define the set T^* that computes the path with the minimum transmission delay from s_h to $\forall c_d \in C$ in Eq. (9).

$$T_{p(s_h,c_d)} = \min(\{t_{e(s_h,s_i)}, t_{e(s_i,s_j)}, \dots, t_{e(s_k,s_d)}\}) \tag{7}$$

$$T_{P(s_h,c_d)} = \min(\{T_{p_1(s_h,c_d)}, T_{p_2(s_h,c_d)}, \dots, T_{p_q,(s_h,c_d)}\}) \tag{8}$$

$$T^* = \min_{\forall c_d \in C} \{T_{P_{(s_h,c_d)}}\} \tag{9}$$

To minimize the link load on overloaded links, we define Eq. (10) to compute packet-in count on every edge that exists between source switch s_h and destination controller c_d, and Eq. (11) is defined to compute the minimum flow count on set of all paths that exists between switch s_h and c_d. Set L^* in Eq. (12) represents the path that exists between s_h and all controllers $c_d \in C$ with the minimum flow count from s_h to controller c_d.

$$L_{p(s_h,c_d)} = \min(\{l_{e(s_h,s_i)}, l_{e(s_i,s_j)}, \dots, l_{e(s_k,s_d)}\}) \tag{10}$$

$$L_{P(s_h,c_d)} = \min(\{L_{p_1(s_h,c_d)}, L_{p_2(s_h,c_d)}, \dots, L_{p_q,(s_h,c_d)}\}) \tag{11}$$

$$L^* = \min_{\forall c_d \in C} \{L_{P_{(s_h,c_d)}}\} \tag{12}$$

We define the set h that determines the hop count of a path from the source switch s_h to destination controller c_d in Eq. (13), whereas Eq. (14) represents the path with the minimum hop count among all the available paths that exist between s_h and c_d. To represent the minimum hop count from s_h to $\forall c_d \in C$, we define the set H in Eq. (15)

$$h_{p(s_h,c_d)} = |p_s, (s_h, c_d)| \tag{13}$$

$$H_{P(s_h,c_d)} = \min(\{h_{p_1(s_h,c_d)}, h_{p_2(s_h,c_d)}, \dots, h_{p_q,(s_h,c_d)}\}) \tag{14}$$

$$H^* = \min_{\forall c_d \in C} H_{P(s_h,c_d)} \tag{15}$$

In our target scenario, since a switch is managed by multiple controllers and each configuration message is sent by multiple controllers for achieving consensus among the controllers, the controller-to-controller path for communication should have the maximum reliability value and minimum delay for the controllers mapping to the switches. This will result in the lowest communication overhead in the consensus process. Thus, we define a first constraint function Φ^d that determines the path from $\forall\{c_d, c_f \in C\}$ with the minimum transmission delay. Function Φ^r determines the path with the maximum reliability among all $\forall\{c_d, c_f \in C\}$. All the symbols used in our proposed approach are included in Table 3. After formulating our target scenario, we describe in the following section our proposed NSGA-II-based algorithm.

$$\Phi^r = \max_{\forall\{c_d,c_f \in C\}} \{R_{P_{(c_d,c_f)}}\} \tag{16}$$

$$\Phi^d = \min_{\forall\{c_d,c_f \in C\}} \{T_{P_{(c_d,c_f)}}\} \tag{17}$$

4. NBFT-SDN: The proposed algorithm

We propose the new NBFT-SDN technique, a Non-dominated Sorting Genetic Algorithm II (NSGA-II)-based algorithm to perform efficient mapping of switches to multi-controllers such as the network tolerates at least f byzantine faulty controllers. The goal is for each switch to be mapped to $3f + 1$ controllers, such as the path from a switch

Table 3
Symbols description.

Symbols	Description
G	Undirected Network Graph
S	Set of Switches i-e $S = \{s_1, s_2, \ldots s_m\}$
C	Set of Controllers i-e $C = \{c_1, c_2, \ldots c_o\}$
E	Set of edges i-e $E = \{e_1, e_2, \ldots e_p\}$ where $e_i = e(s_j, s_k)$
P	Set of paths from source switch s_h to destination controller c_d
R_P	Set representing maximum reliable path in set P
T_P	Set of path with minimum transmission delay in set P
L_P	Set of path with minimum flow count in set L
R^*	Maximum reliable path from s_h to c_d
T^*	Path with minimum transmission delay from s_h to c_d
L^*	Path with minimum hop count from s_h to c_d
H	Minimum hop count of path in set P

to its corresponding controller has maximum reliability, minimum transmission delay, minimum load over the link and minimum hop count, as indicated in Section 3.2. Therefore, our target problem is a multi-objective optimization problem.

NSGA-II is a multi-objective optimization algorithm that is commonly used to solve problems with multiple, often conflicting, objectives like the ones described in Section 3.2 on Problem Formulation, where Eqs. (14) and (17) are minimization objectives and Eq. (16) includes a maximization objective. NSGA-II is an evolutionary algorithm to sort individuals on the basis of dominance. A solution is said to be dominating other if it is better in at least one objective and not worst in any remaining objectives. Sorting of individuals in this criteria forms Pareto-optimal fronts where set of best solution sorted in first non-dominated front, the second front contains the next best non-dominated solutions and so on.

Fig. 2 shows the complete integration of NBFT-SDN to the data and distributed control plane. NBFT-SDN is working at the application layer of the SDN architecture. The Read Topology module uses link layer discovery protocol (LLDP) [34] to learn underlying network topology, link delays, and link states to construct a global view of the entire network. This information is inputted to our proposed NSGA-II algorithm where each individual is evaluated based on the indicated objectives and constraints, as described in Section 4. The Population is refined repeatedly using non-dominated sorting, crossover and mutation operators until the optimal switch-controller mapping is obtained. Each controller in the Control plane gets the list of its assigned switches, which is the outcome of the proposed NBFT-SDN solution running at the application layer. At this point, the control plane is configured and can accept the data plane requests (Packet-In messages) from their assigned switches. When a client request is received by a switch, it will forward it to its $3f+1$ controller group. For each switch, there would be a controller cluster that processes the client request. Selection of the controller in the $3f+1$ cluster is an output of NBFT-SDN algorithm, meeting all the objectives and constraints discussed in Section 4.2 and Section 4.3, respectively. For every data plane request, Synchronization takes place between $3f+1$ controllers to maintain state consistency. In NBFT-SDN we obtain reliable switch-controller mapping, unlike for instance in the approach [9] where the synchronization process may fail if one malicious node is present among the $3f+1$ nodes. In the last step, after successful synchronization process, the controller updates their internal state and sends a computed response to the data plane switch. The switch compares the control messages to commit the configuration response to the switch.

Fig. 3 summarizes the steps of the proposed solution NBFT-SDN. To achieve optimal Switch-controller mapping. This algorithm considers the Graph (G) nodes (*i.e.* controller and switches) and edges with associated reliability and delay attributes. This information is passed to Step 1 of the algorithm where a random population is generated. The Random population represents switch mapping to set of $3f+1$ controllers against each switch. Algorithm 1 discusses the generation

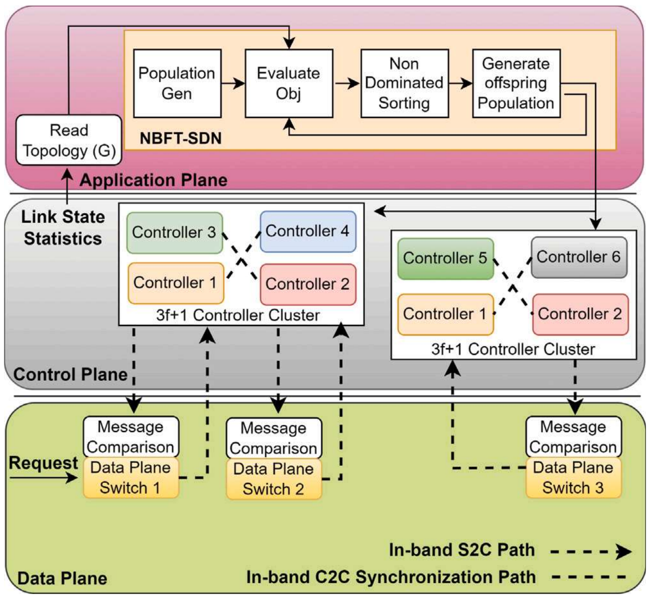


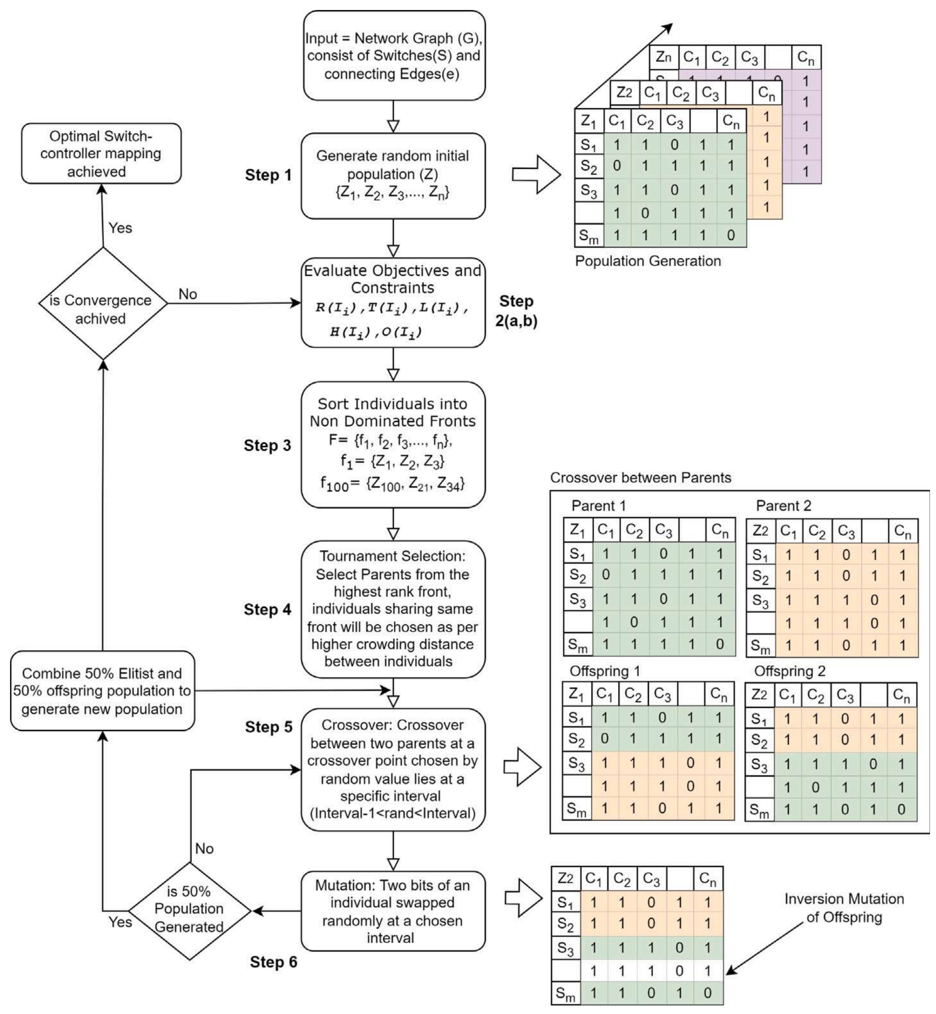
Fig. 2. NBFT-SDN Architecture.

mechanism in detail. Step 2 takes the population and computes the fitness of every individual($I_i \in Z$) as per objective functions and constraints, formulated in Section 4.2 and Section 4.3. These computed fitness scores are utilized by step 3 of the algorithm where every individual is sorted in non-dominated fronts represented as F. All the individuals placed in the first front f_1 if they are not dominated by any other solution in the population, similarly, solutions sorted in the second front f_2 are dominated by at least one objective by the solutions placed in the first front. Algorithm 2 explains this process in detail. Step 4, illustrated Fig. 3 selects individuals(parents) from high-rank fronts for cross-over operation. If it is required to choose individuals from the same front, then individuals with the highest crowding distance relative to the other individuals are selected.

Cross-over operation takes place in step 5 where new offspring are generated by cross-over between two parent individuals. We use our modified cross-over operator that eliminates the chances of offspring being infeasible. This operator enhances the proposed solution to explore the search space to find the best solution. Step 6 introduces the mutation operator, which is based on inversion mutation. The procedure selects two random bits in an individual and swaps their places. The operator enhances the search space exploration strategy of NBFT-SDN. We generate 50% offspring population and combine it with 50% elitist population selected by non-dominated sorting in step 3 to generate a new population. The proposed solution NBFT-SDN iteratively improves the population until no further improvement is observed. Next, we describe the step-by-step working of our NBFT-SDN algorithm as a proposed solution to our problem.

4.1. Step 1: Random initialization of population

We represent our input to the NSGA-II algorithm in the form of a matrix Z of size $m \times n$ where m number of rows correspond to the switches and n number of columns correspond to controllers, as given in Fig. 4. This matrix represents our binary search space. More specifically, a matrix $Z = (0|1)^{m\times n}$ represents particular switch mapping to required number of controllers. One particular individual I shown in Fig. 4 represents one possible solution from entire search space, where "1" in a row s_1 and column c_1 means switch s_1 is mapped to controller c_1, and "0" indicates in row s_3 and column c_3 that switch s_3 is not mapped (*i.e.* controlled) by controller c_3.



Input = Network Graph (G), consist of Switches(S) and connecting Edges(e)
Step 1
Generate random initial population (Z) {Z1, Z2, Z3,..., Zn}
Population Generation
Evaluate Objectives and Constraints R(Ii),T(Ii),L(Ii), H(Ii),O(Ii)
Step 2(a,b)
is Convergence achived
No
Yes
Optimal Switch-controller mapping achieved
Step 3
Sort Individuals into Non Dominated Fronts F= {f1, f2, f3,..., fn}, f1= {Z1, Z2, Z3} f100= {Z100, Z21, Z34}
Step 4
Tournament Selection: Select Parents from the highest rank front, individuals sharing same front will be chosen as per higher crowding distance between individuals
Combine 50% Elitist and 50% offspring population to generate new population
Step 5
Crossover: Crossover between two parents at a crossover point chosen by random value lies at a specific interval (Interval-1<rand<Interval)
Crossover between Parents
Parent 1
Parent 2
Offspring 1
Offspring 2
No
is 50% Population Generated
Yes
Mutation: Two bits of an individual swapped randomly at a chosen interval
Step 6
Inversion Mutation of Offspring

Fig. 3. NSGA-II based Switch-Controller Mapping Algorithm.

$$
\begin{array}{c|ccccc|}
 & c_1 & c_2 & c_3 & \cdots & c_n \\
s_1 & 1 & 1 & 1 & \cdots & 0 \\
s_2 & 1 & 0 & 1 & \cdots & 1 \\
s_3 & 1 & 1 & 0 & \cdots & 1 \\
 & \vdots & \vdots & \ddots & \vdots & \\
s_m & 1 & 0 & 1 & \cdots & 0
\end{array}
$$

Fig. 4. Individual representation controller choice for particular switch.

We generate the whole search space by $|S|.\binom{m}{3f+1}$. From this search space, we select a random population of P_{Size}. However, we note that the total search space of Z may have an individual where a switch is mapped to less than $3f+1$ number of controllers. To address this problem, we modify the random selection process of NSGA-II by randomly selecting the initial population of P_{Size} where every switch is mapped to exactly $3f+1$ number of controllers by using some domain knowledge of problem as follows. As a bit pattern representing controller mapping to specific switch must contains $3f+1$ bits as '1' and rest of bits remain '0', because other all combinations are not desirable or considered infeasible. Our this mechanism enforces our initial population into feasible region and reduces the computation time of the algorithm as well. Formally, this mechanism for random population generation of P_{Size} is given in Algorithm 1. Algorithm 1 takes as input the population size P_{Size}, the set of controllers C and the set of switches S. It computes the probability of presence of controller for any switch based on uniformly distributed random number generator. This step is shown in line 6. At line 11, we ensure for each iteration of controller assignment to switch, and the number of controllers obtained should be equal to $3f+1$.

However, if this condition is not met, the algorithm remains in the loop for a new controller assignment until the required number of controllers is assigned to the switch. This algorithm outputs a three-dimensional matrix $Z^{P_{Size}}_{m\times n}$ representing individuals of random initial population. This output is passed on to the next module of NSGA-II as given in Fig. 3, *i.e.* calculate the Objective Functions, as described in Section 4.2.

4.2. Step 2a: Calculate the objective functions

As in our target problem, a switch s_h is mapped to $3f+1$ controllers, for switch-to-controller communications, there can be k number of finite paths. A path can have different values of the link state information like the values of link reliability, transmission delay and hop count. As

Algorithm 1 Random Population Generation Algorithm

```
Require: $C, S, P_{Size}$ ▷ set of Controllers, Switches and Population Size
Ensure: $Z^{P_{Size}}_{m \times n}$ ▷ Random Population
  for each $n$ in $P_{Size}$ do
    for each $x$ in $S$ do
      for each $y$ in $C$ do
        while $(|Z^n_{(x,\forall y)}| \neq 3f+1$ and $Z^n_{(x,y)} \neq 1)$ do
          $rnd \leftarrow rand()$
          if $(rnd < 3f + 1/Cont)$ then
            $Z^n_{(x,y)} \leftarrow 1$
          else
            $Z^n_{(x,y)} \leftarrow 0$
          end if
          if $(y \equiv |C|$ and $|Z^n_{(x,\forall y)}| < 3f+1)$ then
            $y \leftarrow 0$
          end if
        end while
      end for
    end for
  end for
```

described in Section 3.2, we model objective functions represented in Eq. (5) - Eq. (11) to select the most suitable controller that optimizes all our objectives and constraints from switch $\forall s_h \in S$ to $\forall c_d \in C$. These objective functions compute all the required number of controllers to map each switch to the $3f + 1$ number of controllers. We map the objective functions described in Section 3.2 to the objective functions of NSGA-II as follows. After selecting a random population of size P_{Size} in Section 4.1, we compute the objective functions for each individual element of the population in our proposed NBFT-SDN as follows.

The first objective is to compute the minimum delay objective function represented as $\mathscr{T}(I_i)$ for the individual I_i in the random population of size P_{Size} from each switch s_h to its corresponding controllers $c_d \in C$ for each individual I_i as given in Eq. (18).

$$\{\mathscr{T}(I_i) : \forall \mathrm{i} = 1 \text{ to } P_{Size}\} = \{\{T_{P_{(s_h,c_d)}}\}.I_i| \begin{cases} 1, & \text{if} I_i^{s_h,c_d} = 1 \\ 0, & \text{otherwise,} \end{cases} I_i^{s_h,c_d} \in Z,$$
$$\forall s_h \in S, c_d \in C, \mathrm{i} = 1 \text{ to } P_{Size}\} \tag{18}$$

Secondly, in order to choose the controllers with the maximum reliability path, we compute the second objective function $\mathscr{R}(I_i)$ representing the reliability value of an individual I_i as given in Eq. (19) by considering the reliability value of the path from a switch s_h to its corresponding controller c_d.

$$\{\mathscr{R}(I_i) : \forall \mathrm{i} = 1 \text{ to } P_{Size}\} = \{\{R_{P_{(s_h,c_d)}}\}.I_i| \begin{cases} 1, & \text{if} I_i^{s_h,c_d} = 1 \\ 0, & \text{otherwise,} \end{cases} I_i \in Z,$$
$$\forall s_h \in S, c_d \in C, \mathrm{i} = 1 \text{ to } P_{Size}\} \tag{19}$$

Eq. (20) indicates an individual that computes the minimum flow count for paths from $\forall s_h \in S$ to controller $c_d \in C$.

$$\{\mathscr{L}(I_i) : \forall \mathrm{i} = 1 \text{ to } P_{Size}\} = \{\{L_{P_{(s_h,c_d)}}\}.I_i| \begin{cases} 1, & \text{if} I_i^{s_h,c_d} = 1 \\ 0, & \text{otherwise,} \end{cases} I_i \in Z,$$
$$\forall s_h \in S, c_d \in C, \mathrm{i} = 1 \text{ to } P_{Size}\} \tag{20}$$

Eq. (21) indicates the controller arrangement that minimizes the hop count of paths from $s_h \in S$ to $c_d \in C$.

$$\{\mathscr{H}(I_i) : \forall \mathrm{i} = 1 \text{ to } P_{Size}\} = \{\{hc_{P_{(s_h,c_d)}}\}.I_i| \begin{cases} 1, & \text{if} I_i^{s_h,c_d} = 1 \\ 0, & \text{otherwise,} \end{cases} I_i \in Z,$$
$$\forall s_h \in S, c_d \in C, \mathrm{i} = 1 \text{ to } P_{Size}\} \tag{21}$$

After computing these objective functions, we consider the constrains (*i.e.* maximum load balancing, minimum delay for controller-to-controller communications, and minimum number of controllers) in Section 4.3.

4.3. Step 2b: Calculation of constraints violation

As we considered a set of constraints defined in Eqs. (16) and (17) in Section 3.2, now we compute the constraints for each individual I_i in the random population of size P_{Size} as follows.

Eq. (22) computes the transmission delay for controller(c_d)-to-controller(c_f) communications, this communication is among $3f + 1$ number of controllers $\forall \{c_d, c_f\} \in C$ mapped to switch s_h.

$$\{\phi(I_i) : \forall \mathrm{i} = 1 \text{ to } P_{Size}\} = \{\{T_{P_{(c_d,c_f)}}\}.I_i| \begin{cases} 1, & \text{if} I_i^{s_h,\forall\{c_d,c_f\}} = 1 \\ 0, & \text{otherwise,} \end{cases} I_i \in Z,$$
$$\forall s_h \in S, \forall \{c_d, c_f\} \in C, \mathrm{i} = 1 \text{ to } P_{Size}\} \tag{22}$$

Similarly, Eq. (23) computes the value of path reliability for the controller(c_d)-to-controller(c_f) among the $3f+1$ number of controllers mapped to switch s_h. Selection of particular controllers are dependent upon the respective bit as 1 in individual I_i, as shown in Fig. 4.

$$\{\phi(I_i) : \forall \mathrm{i} = 1 \text{ to } P_{Size}\} = \{\{R_{P_{(c_d,c_f)}}\}.I_i| \begin{cases} 1, & \text{if} I_i^{s_h,\forall\{c_d,c_f\}} = 1 \\ 0, & \text{otherwise,} \end{cases} I_i \in Z,$$
$$\forall s_h \in S, \forall \{c_d, c_f\} \in C, \mathrm{i} = 1 \text{ to } P_{Size}\} \tag{23}$$

Eq. (24) minimizes the number of controller instances used for mapping all the switches. A fitness function with this constraint will prefer the solutions having the minimum number of controller instances used.

$$\{\phi(I_i) : \forall \mathrm{i} = 1 \text{ to } P_{Size}\} = \sum_{\forall s_h \in S} \{ \sum_{\forall c_d \in C} I_i^{s_h,c_d} | \begin{cases} 1, & \text{if} I_i^{s_h,c_d} = 1 \\ 0, & \text{otherwise,} \end{cases} I_i \in Z,$$
$$\forall s_h \in S, \forall c_d \in C\} \tag{24}$$

Let the set c_m be defined as $\{c_m | I_i^{s_h,c_d} = 1, c_m \in C\}$, where c_m represents the set of all controllers used to map the entire data plan switches, where $\forall s_h \in S$ is a particular individual. We aim to achieve load balancing among the controller set c_m in-terms of equal number of switches assigned to each controller. Since each $c_m \in C$ is mapped to $3f+1$ available controllers, we scale load balancing fraction in Eq. (25).

$$\sum_{\forall c_d \in C} \{ \sum_{\forall s_h \in S} I_i^{s_h,c_d} \} \leq ceil[n(3f+1)/|c_m|]| \begin{cases} 1, & \text{if} I_i^{s_h,c_d} = 1 \\ 0, & \text{otherwise,} \end{cases} I_i \in Z,$$
$$\forall s_h \in S, \forall c_d \in C\} \tag{25}$$

Eq. (23)–Eq. (25) represents our problem as a constrained multi-objective optimization problem. Individuals of the random population P_{Size} can be divided into two categories: (i) if an individual I_i meets all these constraints, it is considered as a feasible one and is added to the set of feasible individuals U_{fesb}. (ii) if an individual, say I_j, violates any of the constraints described in Eq. (23) - Eq. (25), then that individual is added to the set of infeasible individuals U_{infesb}. Most of the evolutionary algorithms including NSGA-II prefer feasible solutions over infeasible solutions in the evaluation process (*i.e.* number of generations) [14]. Therefore, this guides the evolutionary process to search and then to converge into the feasible search space. However, constrained problems have their optimal solutions on the boundaries between the feasible and infeasible search space, as shown in [35]. Motivated by this, we keep a good percentage of infeasible individuals into the offspring population until the optimal solution is found. This is an important aspect introduced in NBFT-SDN and is not present in the NSGA-II algorithm. This approach is based on [35] and helps keep the search space around marginal constraints boundary for finding optimal solutions.

We define a set $CV(I) = \{cv_i, cv_j, \ldots, cv_k\}$ that represents the constraints violation measure for particular individuals, where cv_k represents a violation of the kth constraint. For each constraint, we define the threshold value set $TH = \{th_i, th_j, \ldots, th_k\}$ where th_k is the threshold value of the kth constraint. CV measure is computed as the CV number present for any individual, as shown in Eq. (26).

$$\Theta(I) = \begin{cases} \sum_j \sqrt{cv_j(I_i) - th_j} & \text{if } cv_j > th_j, \forall cv \in CV \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

After computing CV, if an individuals $I_i \in Z$ has $\Theta(I_i) \neq 0$, then I_i is added to the set U_{infesb}. Both of these sets are ranked (in Step 3 in Section 4.4) based on the objectives defined in Section 4.3 and the additional objective (*i.e.* CV) to select the best individuals. We select a proportion of individuals from set U_{fesb} and U_{infesb} into the population with parameter α. If α percent of individuals are selected from the feasible set U_{fesb}, the remaining $(1-\alpha)$ percent of individuals are selected from the infeasible set U_{infesb}. The following step applies the non-dominated sorting on these individuals.

4.4. Step 3: Non-dominated sorting of individuals

In this section, we will sort individuals in non-dominating fronts represented as set $F^{fesb} = \{f_i^{fesb}, f_j^{fesb}, \ldots, f_n^{fesb}\}$ and $F^{infesb} = \{f_i^{infesb}, f_j^{infesb}, \ldots, f_n^{infesb}\}$ where F = $F^{fesb} \cup F^{infesb}$. We obtained 50 percent of the elitist individuals from both fronts with a proportion defined as α and included in the next generation. This is as we have conflicting objectives to optimize, and one solution may perform better for one objective, while others perform better for another. Non-dominating sorting is used to organize solutions into levels or fronts, where each front consists of solutions that are not dominated by any other solution in that front.

Based on the objective functions defined in Section 4.2, individual $I_i \in Z$ is said to dominate individual $I_j \in Z$ if I_i is better then I_j in at least one objective. Each individual is compared with all other solutions. If no other solution dominates it, then it is considered as part of the first front f_1 as shown in $line13$ and $line15$. Similarly, dominance is checked for all other solutions. If they are not dominated, they are included in the subsequent fronts. This process continues until all individuals are sorted in their respective fronts. However, solutions placed in front 1 have rank 1 and best of among all other solutions. Once all solutions are placed in fronts from line 1–21 then we determine individuals for next population set containing at least 50 percent elitist solutions from the current population.

New population set Z is formed by selecting elitist α percentage of feasible individuals from their highest rank fronts and $1-\alpha$ percentage of infeasible individuals from their highest rank fronts (*i.e.* $f_k^{infesb} \in F^{infesb}$). If certain solutions are needed to be obtained from any specific front of the same level, then solutions will be further sorted on the basis of crowding distance shown in line 26 and the individuals with higher crowding distance are included in new population (see line 27 and line 35). Set f_i^r defines remaining r individuals to add in population Z. Note that the crowding distance of set f_i computes the Euclidean distance between the current individual and its neighboring solutions. Assume f_i is sorted based on each objective function, the crowding distance of an individual is computed by assigning the normalized difference between its neighboring solutions fitness values as shown in Algorithm 2.

4.5. Step 4: Tournament selection

In this step, we select the pair of individuals from set F for tournament selection in order to select the best individual as parent. For each pair of individuals, one becomes the winner, and is called parent, if it belongs to a higher rank front or has a higher crowding distance, if both solutions came from equal rank fronts.

Algorithm 2 Non-Dominating Sorting

```
Require: $Z' = U_{fesb} \cup U_{infesb}, P_{Size}$          ▷ Random Population
Ensure: $F = F^{fesb} \cup F^{infesb}$                  ▷ Non dominate fronts
  while $Z' \neq \phi$ do
    $k \leftarrow 0$
    for each $I_i$ in $Z'$ do
      for each $I_j$ in $Z'$ do
        if $I_i \neq I_j$ then
          if $I_j > I_i$ then
            $c \leftarrow c + 1$
          end if
        end if
      end for
      if $c \equiv 0$ then
        if $I_i \in U_{fesb}$ then
          $f_k^{fesb} \leftarrow f_k^{fesb} \cup I_i$
        else
          $f_k^{infesb} \leftarrow f_k^{infesb} \cup I_i$
        end if
      end if
    end for
    $Z' \leftarrow Z' \cap f_k^{\forall\{fesb,infesb\}}$
    $k \leftarrow k + 1$
  end while
  for each $f_i^{fesb}$ in $F$ do
    if $|Z \cup \forall\{I \in f_i^{fesb}\}| < \alpha(P_{Size}/2)$ then
      $Z \leftarrow \forall\{I \in f_i^{fesb}\}$
    else
      $f_i^r \leftarrow CrowdingDistance(f_i^{fesb}, Obj)$
      $Z \leftarrow Z \cup \forall\{I \in f_i^r\}$
    end if
  end for
  for each $f_i^{infesb}$ in $F$ do
    if $|Z \cup \forall\{I \in f_i^{infesb}\}| < (1-\alpha)P_{Size}/2$ then
      $Z \leftarrow \forall\{I \in f_i^{infesb}\}$
    else
      $f_i^r \leftarrow CrowdingDistance(f_i^{infesb}, Obj)$
      $Z \leftarrow Z \cup \forall\{I \in f_i^r\}$
    end if
  end for
```

4.6. Step 5: Crossover

In this step, we apply a single point crossover operator on two selected parents to produce a pair of offspring individuals. We define crossover point to chosen at interval $i \in \{1, \ldots, |S|\}$ such that random number r lies between interval $\frac{i-1}{|S|} < r \leq \frac{i}{|S|}$ where each interval has a width represented as $w = 1/|S|$. This method of choosing random point at interval will eliminate chances for the offspring individual to become infeasible.

4.7. Step 6: Mutation

We define the mutation operator as an inversion of bits associated with a randomly chosen switch s_h corresponding to the interval $i \in \{1, \ldots, |S|\}$. A random number r is generated such that it lies between $\frac{i-1}{|S|} < r \leq \frac{i}{|S|}$ where each interval has a width represented as $w = 1/|S|$. After chosen switch, two bits corresponds to the controller mapping are swapped.

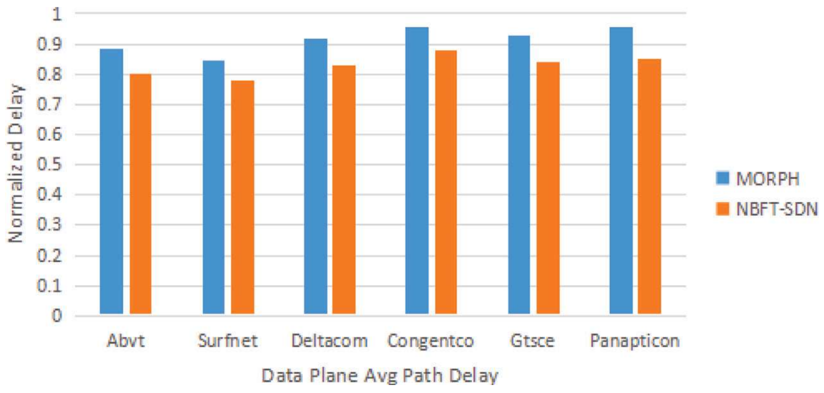


Fig. 5. Average Path delay from Switches to $3f+1$ Controllers.

Algorithm 3 Sorting of Individuals in front f based on Crowding Distance

```
Require: $f_i$, $\Delta = \{\mathscr{T}(I_i), \mathscr{R}(I_i), \mathscr{H}(I_i), \mathscr{L}(I_i)\}$   ▷ $i^{th} front, Allobjectives$
Ensure: $f' = I_i > I_j > .. > I_n$   ▷ sorted individuals based on higher crowding distance $i \leftarrow 1$
1: for each $obj_i$ in $\Delta$ do
2: Sort($obj_i(f_i)$)
3:     $d_{I_i}^{obj} \leftarrow \infty$, $d_{I_n}^{obj} \leftarrow \infty$, $i \leftarrow i+1$
4:     for each $I_i$ in $f$ do
5:         $d_{I_i}^{obj} \leftarrow d_{I_i}^{obj} + (fit(I_{i-1}) - fit(I_{i+1}))/(max(obj_i) - min(obj_i))$
6:     end for
7: end for
8: $f' \leftarrow sort(d_{\forall\{I_i \in f_i\}})$
```

5. Performance evaluation

In this section, we first describe the evaluation metrics used to quantify the effectiveness of the proposed approach. We used real network typologies of different sizes from the internet zoo-topology [15] and Panopticon network [16] summarized in Table 4. Links are labeled with sampled data [17] obtained according to the used topology size. We simulated our algorithm using the Python Networkx [36] library on a Ubuntu16.04 machine with 16 GB RAM.

5.1. Evaluation metrics

We use the following metrics to compare our proposed algorithm NBFT-SDN with the existing solutions Pri-Backup [8] and MORPH [9].

- Data plane delay: It measures the average delay incurred on the paths from all the data plane switches to their corresponding $3f+1$ controllers. Among all delay components (*i.e.* processing, queuing, transmission and propagation), the transmission delay is most relevant in this context, as it depends on the link bandwidth and the packet size. As the transmission delay is different for different links because the bandwidth of the links are different, we select the path whose available bandwidth is maximum. This will lead to the shortest transmission delay. The transmission delay is considered in this approach as done in other existing approaches for BFT in SDN like MORPH [9] and Pri-Backup [7,8] on byzantine fault tolerance, and for general SDN [28].
- Data plane reliability: It is a measure of the average reliability level of the path from all the data plane switches to their corresponding $3f+1$ controllers. We compute the reliability of links by technique given in [12], reliability values range between 0 and 1.

Table 4
Topology nodes and size.

Topology	No. of nodes	No. of edges
Abvt	22	68
Surfnet	49	68
Deltacom	113	161
Gtsce	149	193
Cogentco	197	243
Panopticon	415	570

- Link load by BFT traffic: This parameter measures the effectiveness of minimization of link load by increased traffic load introduced by making network byzantine fault tolerant.
- Controller load balancing - Coefficient of Variation (CoV): It computes the spread of data around its mean, in our context, we have switch mapping to $3f+1$ controllers. We computed it as a fraction between the average sum of the absolute difference between actual switch mapping of controllers (x_i) to the mean value (μ) as shown in Eq. (27). A ratio near to 0 means that there is an even load distribution among controllers. We scale CoV between 0 and 100 for better visualization.

$$CoV = \frac{\sum_{i=1}^{3f+1} |x_i - \mu|}{(3f+1)\mu} \tag{27}$$

- Switch-Configuration Delay: It is measured as the total delay experienced by all data plane switches *i.e.* time to reach packet-in to distributed control plane for consensus and configuration committed by the data plane.
- Re-Synchronization Delay against link failures: It is a measure of synchronization process failure in case of link failures. Our proposed approach NBFT-SDN considers link reliability as a parameter to find an optimal mapping. We simulated data plane traffic and measured the delay incurred due to synchronization failure.
- Packet Congestion Delay experience: This metric measures congestion level induced by mapping switch to $3f+1$ controllers. We allocate link capacity randomly and measure the data rate on each link to compute the link utilization.

Fig. 5 shows the comparison of data plane delay for both our proposed solution NBFT-SDN and MORPH, computed by calculating the delay from every source switch to its $3f+1$ mapped controllers for each packet_in request. We measure this delay by running both algorithms on networking topologies of different sizes. In all experiments, our proposed solution NBFT-SDN finds the paths with shorter delays as compared to the existing solution MORPH.

First, NBFT-SDN considers the transmission delay as an objective and selects the path with the minimum transmission delay. Second, our

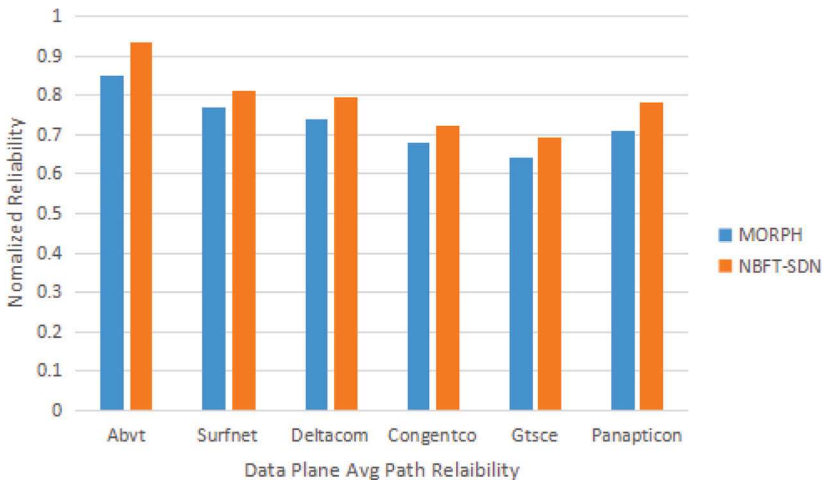


Fig. 6. Average Path Reliability from Switches to $3f+1$ Controllers.

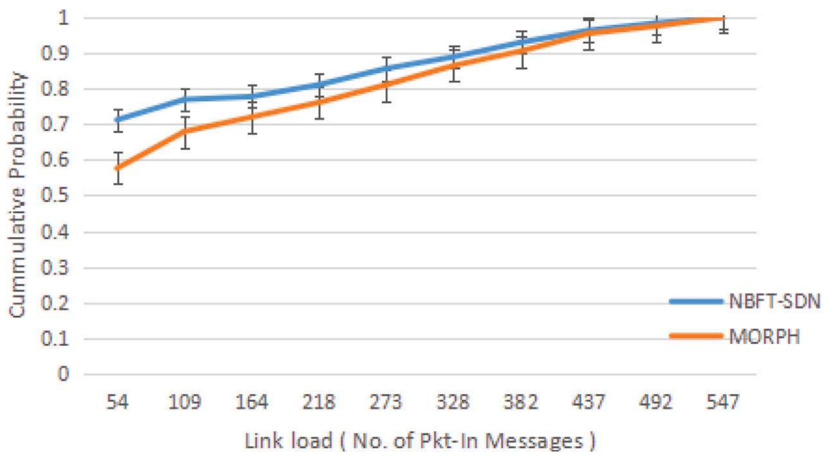


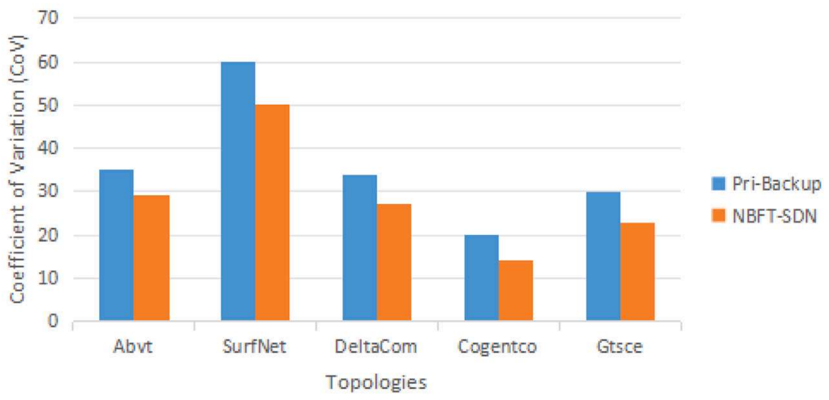
Fig. 7. Link load under BFT Operation.

proposed solution NBFT-SDN also considers the hop count in mapping the switch to its corresponding controllers. This also helps in considering the shorter path for switch to its corresponding controllers. This is as the proposed algorithm finds the optimal solution by optimizing the multi-objective fitness function. It is observed that the obtained switch-controller mapping has significantly reliable paths to carry data plane messages as shown in Fig. 6. Its reason is that our proposed solution NBFT-SDN computes the mapping of switches to controllers such that the path switch-to-controller path has the maximum link reliability. By considering a more reliable path for mapping a switch to its corresponding controllers, it reduces the chances of failure and then subsequently its recovery in case of link failure.

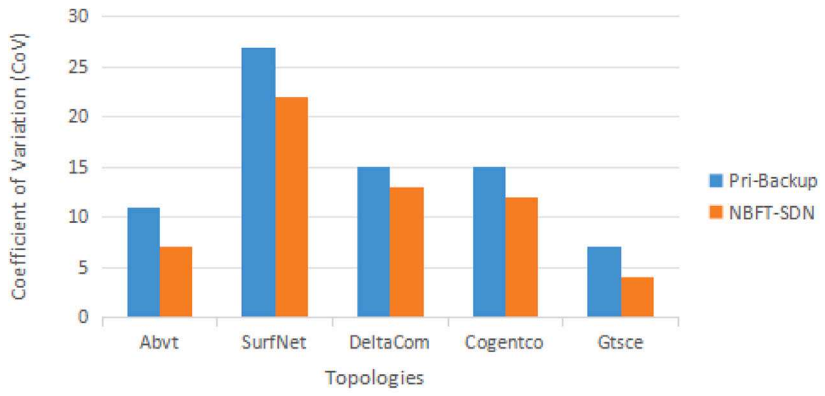
To handle BFT, a significant traffic and computation overhead is incurred both at the data plane and control plane because it introduces the communication of a data plane device (*i.e.* the switch) with a number of replicated control plane, and the communication required among controllers for achieving the synchronization. Therefore, we measure the load of packet-in messages from every switch to its mapped controllers. It has been observed that our proposed solution NBFT-SDN incurs 14% less traffic load compared to the existing approach proposed MORPH [9]. This is because NBFT-SDN considers the objective function of minimum traffic among the controllers in Eq. (12) while computing the mapping of switches to controllers and the controllers deployment.

Fig. 7 shows a comparison of the Cumulative Density Function (CDF) of link traffic load of our proposed solution NBFT-SDN and the existing solution MORPH [9]. Moreover, our proposed approach NBFT-SDN considers the shortest path for communication among the controllers. Further, when we use reliability as the objective function to find the path with maximum reliability, our approach performs better than MORPH [9] because it avoids less reliable links and unnecessary packet re-transmission that contributed to the link load.

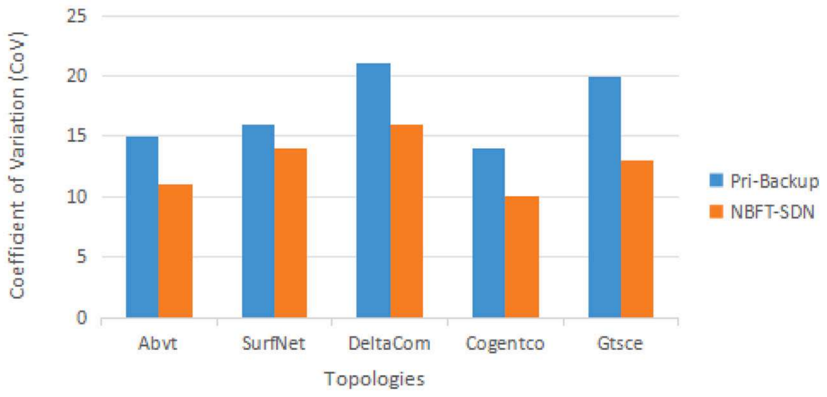
We also evaluate the performance of our proposed approach in terms load balancing among the controllers in Fig. 8. We measure controller load balancing as the spread of controller mapping around the mean value of assigned switches to $3f+1$ controllers. This parameter ensures a fair accounting of load balancing among controllers. For example, if all of the controller in set of $3f+1$ controllers is mapped with an equal number of switches then we compute CoV as 0, closer the ratio to 0 is considered as fairer load distribution. If a controller is more overloaded, then it will introduce the delay. In Fig. 8(a), 8(b) and 8(c) we computed CoV for three different fault tolerance settings: $f = 1$, $f = 2$ and $f = 3$, respectively. Due to fact, that, to tolerate more byzantine faults, we need more number of controllers to map data plane switches. For example for $f = 3$, we need exactly 10 controller for every switch in the data plane to tolerate 3 malicious controllers, hence it will require fair load distribution across control plane. In all experiments, our proposed approach finds better load distribution even when we have a larger topology in the experiment. The reason that we consider this load balancing metric is that, by including a specific constraint in Section 4.3, we evaluate each solution in terms of number of constraint violations.



(a) f = 1



(b) f = 2



(c) f = 3

Fig. 8. Controller Load Utilization: Coefficient of Variation.

In Fig. 9, we compare the CDF of switch configuration delay of our proposed solution NBFT-SDN against the existing solution MORPH. This delay is the time experienced by a switch by sending configuration message to $3f + 1$ controllers, plus the time to compute the response at the controllers and plus the time required by the controllers to synchronize their state with each other, and plus the time required to commit that response by the controllers to target switch. Our proposed approach finds the paths with the minimum delay between switch-to-controller, and controller-to-controller. Additionally, the reliability of the selected path is maximum, and so our mechanism prevents packet loss due to link failures. Third, our load balancing among controllers is the best, as almost equal queuing delay at the $3f + 1$ controllers is achieved. Minimizing the delay for controller-to-controller communication is a critical component to minimize the controllers consensus time that leads to over all switch configuration delay. This optimization has improved the throughput of entire network by computing configuration response in less time. Fig. 9 shows that there is 90% probability that the switch configuration delay computed by NBFT-SDN is 14.8msec whereas it is 15.8msec in the case of MORPH.

In our target scenario, we are computing a reliable path from the data plane switch to $3f + 1$ controllers, and also between controllers for a reliable synchronization process to take place. In the MORPH

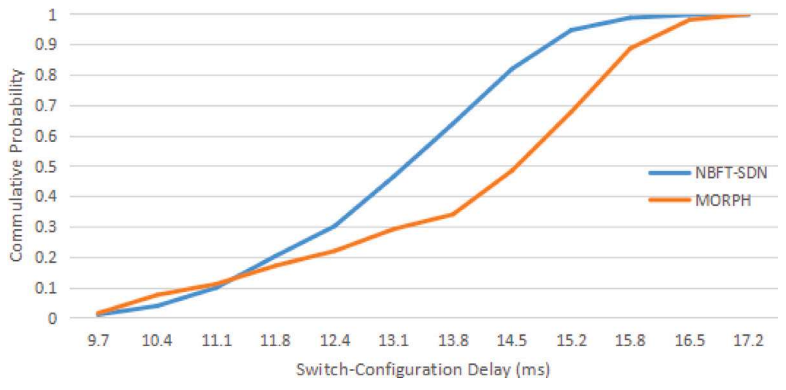


Fig. 9. Switch configuration delay.

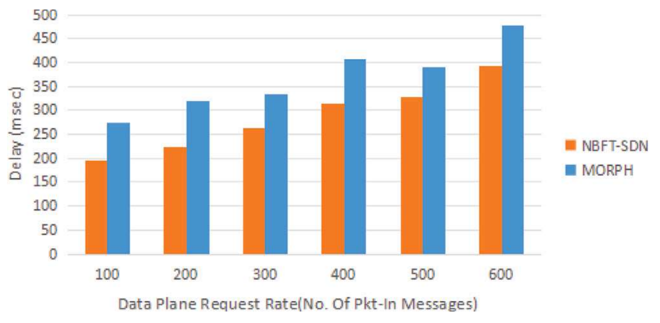


Fig. 10. Synchronization delay in case of link failure.

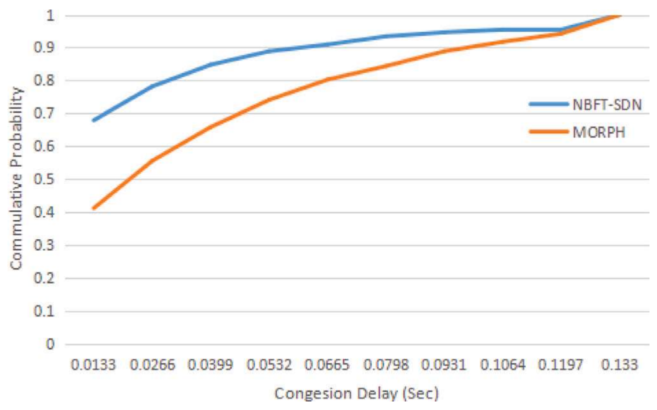


Fig. 11. Per packet congestion delay.

approach [9] the paths between switch to controllers and between controllers are not reliable. We also computed the synchronization overhead by simulating the data plane traffic randomly and showed the additional delay incurred due to the loss of packets and the effect on the synchronization process. Fig. 10 illustrates a cumulative delay comparison of our proposed approach NBFT-SDN and MORPPH [9]. Data plane requests are randomly generated in the network and the number of synchronization failures were counted. For every counted value there is an extra configuration delay including time taken by the synchronization process. Our approach NBFT-SDN considers the reliability of the paths and has experienced less packet loss during the synchronization.

The performance of the proposed approach NBFT-SDN is evaluated by measuring the packet delay experienced on the network links. We randomly vary the link capacity($Cp_{e(i,j)}$) from 1 kbits/s to 5 kbits/s on the topology links under test. The average packet size(pkt_{size}) is 1400 bytes, a little less than the Maximum Transmission Unit (MTU) size of 1500 bytes.

We plot the CDF of the congestion delay experienced by packets on the particular link. Fig. 11 compares the proposed approach NBFT-SDN and MORPH. Packets experience about 26% less congestion when our approach is employed. This is because we model link load as one of the parameters of the multi-objective optimization. In our proposed approach NBFT-SDN, the congestion value is 0.0133 s or less on 69% of the links, whereas, in the approach MORPH, this value is measured on 42% of the links only. Traffic load induced by mapping on a large number of controllers $(3f + 1)$ to tolerate byzantine faults can overload links. To overcome this problem, it is important to avoid congested links. This improves the network throughput in terms of synchronization and switch configuration delay.

6. Discussion

In this section, we present some challenging scenarios related to our research problem. First, we describe the link failure scenario, that can impact the performance of the proposed approach NBFT-SDN. Second, we highlight the synchronization overhead introduced by BFT in the context of SDN. Third, we explain the effectiveness of the proposed approach to reduce the re-transmission of packets due to packet lost during link failure.

6.1. An example of a link failure scenario

Note in Fig. 1 from Section 3 the BFT problem in SDN, after executing the NSGA-II algorithm, the switch sw(i) is mapped to the controllers $\{C_1, C_3, C_4, C_6\}$ as shown in Fig. 12. Further, suppose a link, say $e(s_i, s_l)$ of the path p_a (i-e $p_a, (s_i, c_4) = \{s_i, s_l, c_4\}$) connecting a switch, sw(i) in this case, to a controller C4, fails, then the switch sw(i) can connect to C4 via another path p_b or p_c, as shown in Fig. 12. The path p_c is better than p_b in terms of delay from switch sw(i) to sw(l), whereas path p_b is a more reliable path between sw(i) and sw(l). Although path p_b seems better from the sw(i) perspective, if we execute the NSGA-II, it will result in the following optimum solution $\{C_1, C_2, C_3, C_5\}$. So there are two alternatives as either we should only recompute the path from the sw(i) to C4 without recomputing the re-mapping of all switches to all controllers using NSGA-II algorithm, or we should recompute the re-mapping of all the switches to all the controllers from scratch using the NSGA-II algorithm. Both of these alternatives have their own advantages.

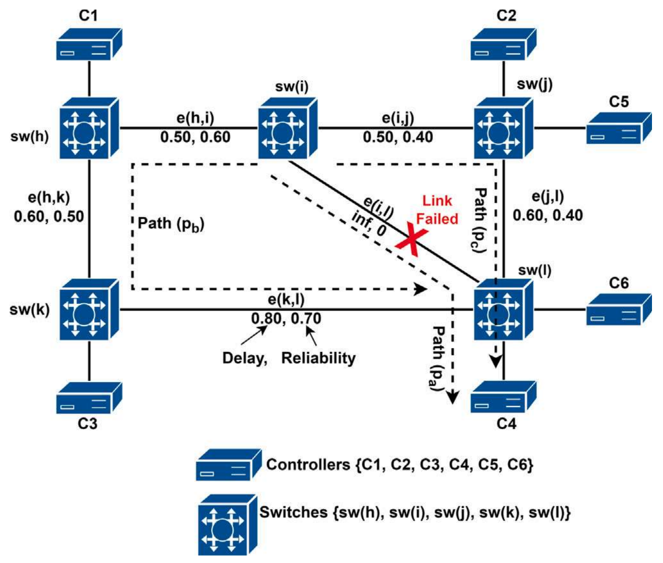


Fig. 12. A multi-controller example topology(Link Failure).

In the former approach, recalculating a new path from the sw(i) to $C4$ would be computationally less expensive, but may not give the global optimum mapping of switches to controllers with the given objectives and constraints. In the later alternative, all the switches will be optimally mapped to the controller based on the given objectives and constraints using the NSGA-II algorithm. However, this introduces a significant computation delay. Additionally, if during this re-computation process, the switch $sw(i)$ migrates to a new cluster *e.g.* C_2, C_5 are new controllers in cluster $\{C_1, C_2, C_3, C_5\}$ then the state will not be synchronized among all controllers. So, in this case, the switch-to-Controller migration cost in terms of number of transmissions in the network can be significant. For example, migrated sw(i) may have large number of data plane requests in comparison with the previous optimal controller mapping *i.e.* $\{C_1, C_3, C_4, C_6\}$.

Moreover, the switch-to-controller mapping is a challenging problem for variable link delays, link failures, and data plane traffic. First, link congestion is a common problem in computer networks and it is observed on the router/switch interconnect. Second, link failures can disrupt the byzantine fault tolerance in terms of a failure of the synchronization process that leads to excessive packet re-transmission to tolerate more than one byzantine fault. Third, data plane traffic is also highly variable in production environments where the number of users and their usage traffic have diverse service level agreements. The dynamic nature of these three parameters may degrade the performance of the network if switches are statically mapped to controller using an optimization algorithm. However, obtaining new optimal solutions after any change in these parameters will determine frequent recomputing of the switch-to-controller remapping, and thus it is computationally expensive. Moreover, such frequent recomputing of the mapping of switches to controllers introduces a control plane scalability problem.

6.2. Controllers synchronization overhead

Our proposed solution maps each switch to $3f+1$ controllers, where f is the number of byzantine faults being tolerated. Each data plane request packet is forwarded to $3f+1$ controllers, where synchronization between controllers takes place in a series of steps, as shown in Fig. 13 and already used by the existing approach [19]. In the first phase (Request), the data plane request is being forwarded to $3f + 1$ controllers. In the second stage (Prepare), every controller multicast its received request to other controllers to enforce the execution order of the client request. In the Commit phase, controllers independently execute client requests to find the responses and multicast the computed responses to other controllers in the 3f+1 cluster. In the last stage, each controller sends its response to the client to commit the configuration response. For the configuration message to be successfully committed to the switch, it is necessary to find at least $2f + 1$ identical messages in the reply phase. Thus the number of tolerated byzantine faults must be f. This process attempts to ensure consistency and synchronization among the controllers. However, some inconsistency due to link failure may still occur as discussed next.

To maintain the synchronization between $3f + 1$ controllers, all controllers must have the same global view of the network to compute an identical response. Link failure can cause the network's global view to be inconsistent. For example, the control plane is in the process of computing a response for resource reservation requests from the data plane. Assume the path p_x is their consistent output, however, due to the link failure of the same path, one or a few of the controllers may not get this change in time. As a result, those controllers which have received the link failure event timely compute a different path (*e.g.* path p_y) for resource reservation. However, path p_x will be computed by a controller that did not receive a link failure event within the threshold. Since in our target scenario we have $3f + 1$ controllers mapped to each switch, We still compute consistent response if one controller may not get a link failure message because $2f + 1$ controllers remain consistent as shown in Fig. 13. However, if more than one controller out of four controllers $(3f + 1, where f = 1)$ did not receive a failure event then the consensus will not be reached for path computation. In Fig. 13 we have Controller 4 as a malicious controller (or maybe in an inconsistent state) that may not get a link failure message and compute a malicious response. Though this link failure can cause an inconsistency among the controllers, therefore, our proposed approach uses the path with the maximum link reliability for the switch-to-controller communication, and maps the switches to the controllers using the maximum link reliability as one of the objectives along with the other objectives.

6.3. Re-transmission of packets due to link failure

Re-transmission occurs in the network frequently due to the loss of packets during communication. The following example explains link failure and re-transmission of data in the context of BFT in SDN. Assume, we have already computed optimal switch-to-controller mapping using the NSGA-II algorithm in Fig. 1 from Section 3 where switch sw(i) is mapped to $3f + 1$ controllers (*i.e.* $\{C_1, C_3, C_4, C_6\}$). To compute a response against a data plane request, sw(i) must communicate with all its mapped controllers via a reliable path. Suppose, the link between sw(i) and sw(l) is failed (*i.e.* $e(i, l)$) during communication as shown in Fig. 12 (Section 6.1). Due to this failure, the data packets are lost at the data link and routing layer. In SDN, switches and controllers communicate using OpenFlow protocol [37]. OpenFlow is an application layer protocol that uses the Transmission Control Protocol (TCP) for the reliable delivery of packets. The loss of packets due to the link failure is detected at the TCP layer of the switch/controller, which re-transmits them. This re-transmission of data packets occurs frequently due to link failures and can overload other paths.

Moreover, for BFT in SDN, each switch is mapped to minimum $3f+1$ controllers. In this scenario, links are shared to carry data packets from other switches and controllers as well. Link failure in such cases causes link congestion [38] and impacts the available bandwidth of other available links. To reduce the unnecessary re-transmission of data packets, we handle this situation in two ways. First, we used link reliability as an objective function defined in Section 4.2 to maximize it, so that we find a path for switch-to-controller communication, and between controllers with the maximum reliability value. This ensures the lowest packet loss, leading to the fewest re-transmission of data packets. Second, we reduce the load on congested links by computing link load as

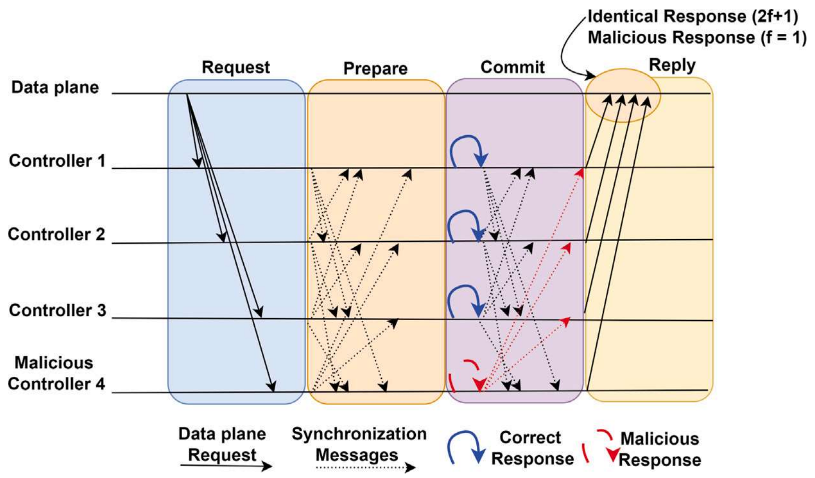


Fig. 13. Controller Synchronization in presence of (f=1) malicious (or inconsistent) node.

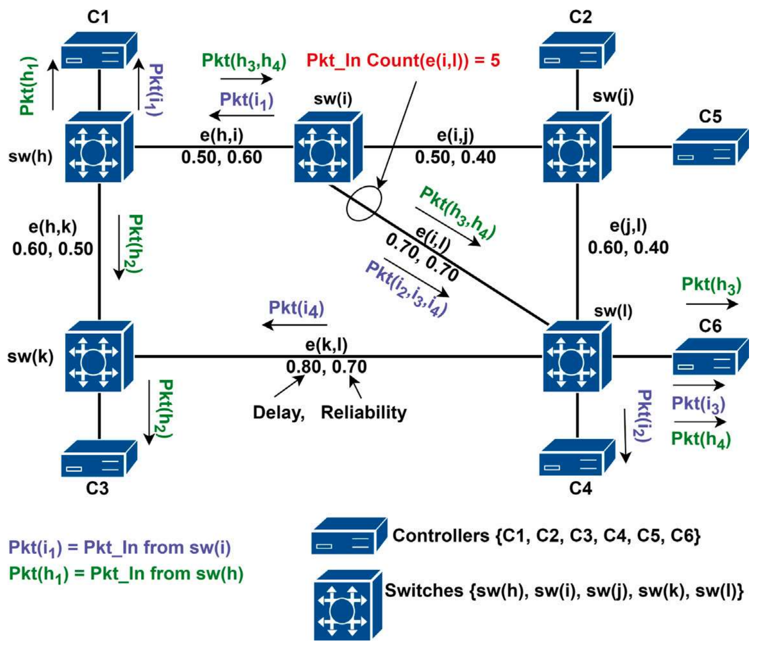


Fig. 14. Link load due to packet-in messages.

an objective function to minimize it. Our proposed solution NBFT-SDN avoids congested links and enables packets to be transmitted through less loaded links. This is because BFT operation involves mapping of data plane switches to a large number of $3f+1$ controllers. Further, synchronization between $3f+1$ controllers increases the number of messages exchanged between switch and controllers to commit a single packet-in response. We explain the computation of link load by an example scenario in Fig. 14.

We obtain the switch-controller mapping using NBFT-SDN and this algorithm iteratively evaluates the individuals by conflicting objective functions and constraints as defined in Sections 4.2 and 4.3. The link load is one of the objectives minimized in NBFT-SDN. The figure shows our concept of excessive link load in the case of making network Byzantine Fault Tolerant, and computing link load over the single link *i.e.* $e(i,l)$. The switches sw(i) and sw(h) are mapped to controller list $\{C_1,C_3,C_4,C_6\}$. The data plane switches such as sw(i) and sw(h) send packets towards all the $3f+1$ controllers, link $e(i,l)$ shows the flow of three packets from sw(i) and two packets from sw(h), making the total count to five on this link. Similarly, we counted all the packet messages on each link and computed CDF function in Fig. 7 (Section 5). This figure shows the percentage of less loaded links in comparison to the alternative approach MORPH [9].

AI algorithms [39–41] are the most appropriate for predicting link reliability dynamically over discrete time intervals before switch-to-controller mapping. In this way we have prior knowledge of link reliability before the switch-to-controller mapping. Further, mapping of all switches to all the controllers using the NSGA-II algorithm based on the given objectives and constraints can be performed in advance before the link failure occurs. So, when the link failure occurs, we remap the switches to controllers for BFT as per the output of the NSGA-II algorithm. This helps save the computation time taken by the NSGA-II algorithm in case of link failure improving the system's response time significantly.

However, despite its advantages, predicting link reliability before switch-to-controller mapping using an AI algorithm has some overhead due to several factors. First, modern AI algorithms [42] require the existence of powerful computational resources like hardware accelerators [43] and Graphical Processing Units (GPU) [44] to train on large data. These accelerators have upfront costs and take significant time to train the model to make accurate predictions. They use parallelism to increase throughput against a CPU that is 100 times slower. Real network topology has hundreds of switches and in order to manage all these switches we need a distributed control plane for scalability and BFT. This type of network contains hundreds of paths that can exist between one source (switch) to a destination (controller). Each path has a distinct reliability value and an AI algorithm must be trained on all such paths with accurate reliability values.

Second, AI algorithms do not have 100% accuracy and may result in inaccurate predictions. For example, in our scenario, an AI algorithm can predict a most reliable link as a less reliable link. Similarly, an AI algorithm can predict a less reliable link as the most reliable. This is termed as False Positive (FP). In this case, our proposed approach will use this link for switch-to-controller mapping by assuming that this link is the most reliable and will not fail. However, as the link is less reliable, so this link will fail. When it fails, then the concerned switch will detect it and then our proposed will compute the remapping the switch-to-controller, and then will reperform the switch-to-controller mapping. During this time of remapping, the performance of switch-to-controller communication will be degraded. This is another effect of employing an AI algorithm. The primary cause of FP and FN in the prediction made by the AI algorithm can be a bias in the training data [45,46]. For example, training data contains imbalanced samples of positive (reliable) and negative (unreliable) paths.

Third, real networks are changing over time for example a reliable link can become unreliable due to congestion or link failure due to hardware malfunction. An AI algorithm [26,42] trained on specific data may recognize this event as per its seen samples during training and predict this path as a reliable one from switch-to-controller. This would result in further loss of data and synchronization packets.

7. Conclusions and future work

In this paper, we proposed a novel approach NBFT-SDN to tolerate f number of faulty controllers in the context of BFT for SDN. Along with parameters used in the existing literature, our proposed approach used some new parameters (*i.e.* link reliability and link load) and used a new AI algorithm (NSGA-II) to solve the multi-objective problem formulated. BFT operation in SDN consumes a significant portion of the bandwidth of data links in case of link failure that leads to data re-transmission, and for BFT control traffic, particularly for the in-band mode of network operation. Since BFT in SDN requires $3f+1$ controllers to map for each switch, we cannot decrease this transmission overhead by BFT algorithm requirement. However, our approach NBFT-SDN avoids less reliable and congested links during the switch-to-controller mapping such that the traffic overhead is reduced in the network. The results showed that NBFT-SDN-based mapping of switches to the controller results in 10% lower delay and up to 10% improvement in the average path reliability for the switch-to-controller communication in comparison with an alternative solution. The proposed approach also performs better in terms of switch load balancing and minimization of the link load by up to 10% and 14%, respectively. Future work will consider the extension of NBFT-SDN in other challenging contexts such as those of vehicular networking, where Switch-Controller mapping is challenging problem for variable link delays and data plane traffic. We would also like to explore the scenario where the traffic load on the link that causes the congestion (due to queuing delay) is predicted using a machine learning algorithm. Based on this predicted traffic load and along with existing objectives and constraints, we would like to compute the mapping of the switches to the controllers using an improved version of the NSGA-II algorithm.

CRediT authorship contribution statement

Waqas Ahmed: Writing – review & editing, Writing – original draft, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Nadir Shah:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Gabriel-Miro Muntean:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

N. Shah wishes to acknowledge the support of Higher Education Commission (HEC), Pakistan via the project no. 20-14884/NRPU/R&D/HEC/2021.

G.-M. Muntean wishes to acknowledge the support of Science Foundation Ireland, Ireland grants 12/RC/2289_P2 (Insight) and 21/FFP-P/10244 (FRADIS).

References

[1] Yassine Maleh, et al., A comprehensive survey on SDN security: threats, mitigations, and future directions, J. Reliab. Intell. Environ. 9 (2) (2023) 201–239.

[2] Bassey Isong, et al., Comprehensive review of SDN controller placement strategies, IEEE Access 8 (2020) 170070–170092.

[3] Murat Karakus, Arjan Durresi, A survey: Control plane scalability issues and approaches in software-defined networking (SDN), Comput. Netw. 112 (2017) 279–293.

[4] Dan Marconett, S.J. Ben Yoo, Flowbroker: A software-defined network controller architecture for multi-domain brokering and reputation, J. Netw. Syst. Manage. 23 (2) (2015) 328–359.

[5] Diego Ongaro, John Ousterhout, In search of an understandable consensus algorithm, in: 2014 USENIX Annual Technical Conference, Usenix ATC 14, 2014, pp. 305–319.

[6] Karim ElDefrawy, Tyler Kaczmarek, Byzantine fault tolerant software-defined networking (SDN) controllers, COMPSAC, in: 2016 IEEE 40th Annual Computer Software and Applications Conference, vol. 2, IEEE, 2016, pp. 208–213.

[7] Purnima Murali Mohan, Tram Truong-Huu, Mohan Gurusamy, Primary-backup controller mapping for Byzantine fault tolerance in software defined networks, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.

[8] Purnima Murali Mohan, Tram Truong-Huu, Mohan Gurusamy, Byzantine-resilient controller mapping and remapping in software defined networks, IEEE Trans. Netw. Sci. Eng. 7 (4) (2020) 2714–2729.

[9] Ermin Sakic, Nemanja Đerić, Wolfgang Kellerer, MORPH: An adaptive framework for efficient and Byzantine fault-tolerant SDN control plane, IEEE J. Sel. Areas Commun. 36 (10) (2018) 2158–2174.

[10] He Li, et al., Byzantine-resilient secure software-defined networks with multiple controllers in cloud, IEEE Trans. Cloud Comput. 2 (4) (2014) 436–447.

[11] Cing-Yu Chu, et al., Congestion-aware single link failure recovery in hybrid SDN networks, in: 2015 IEEE Conference on Computer Communications, INFOCOM, IEEE, 2015, pp. 1086–1094.

[12] Muhammad Ibrar, et al., Reliability-aware flow distribution algorithm in SDN-enabled fog computing for smart cities, IEEE Trans. Veh. Technol. 72 (1) (2022) 573–588.

[13] Yaser Al Mtawa, Anwar Haque, Hanan Lutfiyya, Migrating from legacy to software defined networks: A network reliability perspective, IEEE Trans. Reliab. 70 (4) (2021) 1525–1541.

[14] Kalyanmoy Deb, et al., A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[15] S. Knight, et al., The internet topology zoo, IEEE J. Sel. Areas Commun. 29 (9) (2011) 1765–1775, http://dx.doi.org/10.1109/JSAC.2011.111002.

[16] Dan Levin, et al., Panopticon: Reaping the {Benefits} of incremental {SDN} deployment in enterprise networks, in: 2014 USENIX Annual Technical Conference, USENIX ATC 14, 2014, pp. 333–345.
[17] Muhammad Ibrar, et al., IHSF: An intelligent solution for improved performance of reliable and time-sensitive flows in hybrid SDN-based FC IoT systems, IEEE Internet Things J. 8 (5) (2020) 3130–3142.
[18] Sol Han, et al., Switch-centric Byzantine fault tolerance mechanism in distributed software defined networks, IEEE Commun. Lett. 24 (10) (2020) 2236–2239.
[19] Ermin Sakic, Wolfgang Kellerer, BFT protocols for heterogeneous resource allocations in distributed SDN control plane, in: ICC 2019-2019 IEEE International Conference on Communications, ICC, IEEE, 2019, pp. 1–7.
[20] A. Schrijver, Theory of Linear and Integer Programming, John Wiley & Sons, 1998.
[21] Chien-Fu Cheng, et al., Reaching consensus with byzantine faulty controllers in software-defined networks, Wirel. Commun. Mob. Comput. 2021 (2021) 1–9.
[22] Giovanni Venâncio, et al., VNF-consensus: A virtual network function for maintaining a consistent distributed software-defined network control plane, Int. J. Netw. Manage. 31 (3) (2021) e2124.
[23] Leslie Lamport, The part-time parliament, in: Concurrency: The Works of Leslie Lamport, 2019, pp. 277–317.
[24] Shadi Moazzeni, et al., Improving the reliability of Byzantine fault-tolerant distributed software-defined networks, Int. J. Commun. Syst. 33 (9) (2020) e4372.
[25] Ermin Sakic, Wolfgang Kellerer, Decoupling of distributed consensus, failure detection and agreement in sdn control plane, in: 2020 IFIP Networking Conference (Networking), IEEE, 2020, pp. 467–475.
[26] Munwar Ali, et al., Semantic-k-NN algorithm: An enhanced version of traditional k-NN algorithm, Expert Syst. Appl. 151 (2020) 113374.
[27] Volodymyr Mnih, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[28] Aamir Akbar, et al., SDN-enabled adaptive and reliable communication in IoT-fog environment using machine learning and multiobjective optimization, IEEE Internet Things J. 8 (5) (2020) 3057–3065.
[29] Ermin Sakic, et al., P4BFT: Hardware-accelerated Byzantine-resilient network control plane, in: 2019 IEEE Global Communications Conference, GLOBECOM, IEEE, 2019, pp. 1–7.
[30] Suhail Ahmad, Ajaz Hussain Mir, Scalability, consistency, reliability and security in SDN controllers: a survey of diverse SDN controllers, J. Netw. Syst. Manage. 29 (2021) 1–59.
[31] Songbai Liu, et al., A survey on learnable evolutionary algorithms for scalable multiobjective optimization, IEEE Trans. Evol. Comput. (2023).
[32] Faezeh Pasandideh, et al., Topology management for flying ad hoc networks based on particle swarm optimization and software-defined networking, Wirel. Netw. (2022) 1–16.
[33] Vahid Ahmadi, et al., A hybrid NSGA-II for solving multiobjective controller placement in SDN, in: 2015 2nd International Conference on Knowledge-Based Engineering and Innovation, KBEI, IEEE, 2015, pp. 663–669.
[34] Ahmed Binsahaq, Tarek R. Sheltami, Khaled Salah, A survey on autonomic provisioning and management of QoS in SDN networks, IEEE Access 7 (2019) 73384–73435.
[35] Tapabrata Ray, et al., Infeasibility driven evolutionary algorithm for constrained optimization, Constr.-Handl. Evol. Optim. (2009) 145–165.
[36] Aric Hagberg, Pieter Swart, Daniel S. Chult, Exploring Network Structure, Dynamics, and Function Using NetworkX, Tech. Rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
[37] Birglang Bargayary, Nabajyoti Medhi, SDBlock-IoT: A blockchain-enabled software-defined multicontroller architecture to safeguard OpenFlow tables, J. Netw. Syst. Manage. 32 (4) (2024) 66.
[38] David Franco, et al., Quantitative measurement of link failure reaction time for devices with P4-programmable data planes, Telecommun. Syst. 85 (2) (2024) 277–288.
[39] Yi-Ren Chen, et al., RL-routing: An SDN routing algorithm based on deep reinforcement learning, IEEE Trans. Netw. Sci. Eng. 7 (4) (2020) 3185–3199.
[40] Junjie Zhang, et al., CFR-RL: Traffic engineering with reinforcement learning in SDN, IEEE J. Sel. Areas Commun. 38 (10) (2020) 2249–2259.
[41] Xuancheng Guo, et al., Deep-reinforcement-learning-based QoS-aware secure routing for SDN-IoT, IEEE Internet Things J. 7 (7) (2019) 6242–6251.
[42] Ali Malik, et al., Intelligent SDN traffic classification using deep learning: Deep-SDN, in: 2020 2nd International Conference on Computer Communication and the Internet, ICCCI, IEEE, 2020, pp. 184–189.
[43] Manar Abu Talib, et al., A systematic literature review on hardware implementation of artificial intelligence algorithms, J. Supercomput. 77 (2) (2021) 1897–1938.
[44] Muthukumaran Vaithianathan, et al., Comparative study of FPGA and GPU for high-performance computing and AI, ESP Int. J. Adv. Comput. Technol. (ESP-IJACT) 1 (1) (2023) 37–46.
[45] Chong Zhang, et al., A cost-sensitive deep belief network for imbalanced classification, IEEE Trans. Neural Netw. Learn. Syst. 30 (1) (2018) 109–122.
[46] Fang Feng, et al., Using cost-sensitive learning and feature selection algorithms to improve the performance of imbalanced classification, IEEE Access 8 (2020) 69979–69996.

Waqas Ahmed has been doing Ph.D. from COMSATS University Islamabad, Wah Campus, Pakistan. His interest is in computer networking and SDN.

Nadir Shah received the B.Sc. and M.Sc. degrees from Peshawar University, Peshawar, Pakistan, in 2002 and 2005, respectively, the M.S. degree from International Islamic University, Islamabad, Pakistan, in 2007, all in computer science, and the Ph.D. degree from Sino-German Joint Software Institute, Beihang University, Beijing, China. He was a Lecturer with the Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan, from 2007 to 2008. He is currently a Tenured Professor with the COMSATS University Islamabad, Wah Campus. He has authored several research papers in international journals/conferences, such as the ACM Computing Surveys and the IEEE Communication Letters. His current research interests include computer networks, distributed systems, and network security. He is serving in the editorial board of IEEE Softwarizations, AHWSN, IJCS, and MJCS. He has been serving as a Reviewer for several journals/conferences, including ICC, INFOCOM, WCNC, Computer Networks (Elsevier), IEEE Communications Letters, IEEE Communication Magazine, IEEETransactions on Industrial Informatics, and The Computer Journal.

Gabriel-Miro Muntean (M'04, SM'17, F'23) is Professor with the School of Electronic Engineering, Dublin City University (DCU), Ireland, and Co-Director of the DCU Performance Engineering Laboratory. He has published over 500 papers in top-level international journals and conferences, including 4 authored and 6 edited books. His research interests include quality, performance, and energy saving issues related to rich media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. He is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communications Surveys and Tutorials, and chair and reviewer for important international journals, conferences, and funding agencies.