Fault-Resilience for Bandwidth Management in Industrial Software-Defined Networks

Rutvij H. Jhaveri, *Senior Member, IEEE,* Sagar V. Ramani, Gautam Srivastava, *Senior Member, IEEE,* Thippa Reddy Gadekallu, *Senior Member, IEEE,* Vaneet Aggarwal *Senior Member, IEEE*

***Abstract*—Industrial Cyber-Physical Systems (ICPS) expect assurances of timely delivery of data even during the occurrence of distinct faults. It is key to manage the required bandwidth by providing resilience to link failures and dynamically changing bandwidth requirements. In this paper, we address the aforementioned challenges of managing required bandwidth for traffic flows in ICPS by exploring software-defined networks (SDN). We present a framework coined SDN-RMbw (Software-Defined Networking Resilience Management for Bandwidth), which is a contract-based framework, where the components are bound to bandwidth contracts and a resilience manager. The bandwidth contracts state the bandwidth requirements of traffic flows. With each such contract, a monitor is associated, which is responsible to detect two events, run-time changes and link failures. Directly after receiving the event trigger reports from the monitor, new routes are calculated by a path-finding algorithm. Based on the newly calculated routes, an observer detects whether the contract still satisfies the bandwidth requirements of the traffic flows or the contract gets violated (termed as fault). To provide resilience to such faults in the network, a resilience manager integrated with control logic decides and executes a suitable response strategy (depending upon the severity of the fault). The proposed SDN-based framework thus aims at providing fault-resilience as well as adapting to the network-state changes for the traffic flows. The proposed framework is evaluated using a Ryu SDN controller on a hardware testbed. Our results show that the proposed framework provides enhanced network resilience as compared to baseline mechanisms and improves the success rate up to 21% and bandwidth up to 111 Mbps under distinct network scenarios. Furthermore, extensive experimental emulations on the Mininet SDN tool shows the scalability of the proposed framework.**

***Index Terms*—Fault resilience, bandwidth management, SDN, contracts.**

I. Introduction

The advancements in Computer Science, Information Technology, and communication technologies present a new mindset when building Industrial Cyber-Physical Systems (ICPS) for smart manufacturing. An ICPS typically integrates an industrial process with sensors, actuators, and a computing platform. Such a system requires real-time communication of the cyber components with the industrial components. To provide fault-tolerant communication during events such as dynamic changes in traffic flow requirements and network link failures, the re-examination of the existing communication network architecture and design is imperative [1].

Critical applications in ICPS typically demand lower bounds on the bandwidth of the communication channel to achieve higher throughput [2]. The key challenges in designing such ICPS are: (1) to satisfy different bandwidth requirements of distinct traffic flows between multiple source-destination pairs at run-time and, (2) to improve fault resilience to dynamic changes in bandwidth requirements of flows and to link failures. As conventional network architectures fail to address these challenges, the emerging SDN technology can be exploited to develop reliable and inexpensive solutions. SDN is a novel network architecture that separates the control plane from the forwarding plane which was formerly tightly bound to the forwarding plane in individual network devices [3]. A software-based SDN controller containing control functions can now update the forwarding policies during run-time and thus, provide flexibility, centralized control, and higher network intelligence. Fig. 1 represents an SDN architecture for ICPS. The application program directly communicates with the control plane to get an abstract view of the SDN controller via North Bound Interface (NBI) agents. As a result, the application plane can take suitable decisions via application logic when the network state changes. The data plane consists of network elements as an integrated substantial consolidation of network resources. The presented framework can be used in ICPS to provide uninterrupted services in real-time communication as discussed in Section III.

In this paper, the issue of managing bandwidth requirements of traffic flows using an SDN-based fault-resilient framework is addressed. We complement our previous work [4] to propose the fault-resilient framework to manage dynamic bandwidth requirements of multiple traffic flows in wired networks. We integrate contract-based methodology [4] with the proposed framework to efficiently verify the requirement satisfaction of the traffic flows. The proposed framework can detect distinct events such as dynamic changes in bandwidth requirements and link failures. These events may cause a contract failure (*fault*). When a resilience manager (RM) is reported with such a fault, it immediately responds to the fault by performing suitable operations to manage the bandwidth requirements of the flows. The prototype of the proposed framework is

Corresponding Author: Gautam Srivastava

Rutvij H. Jhaveri is with Dept. of CSE, Pandit Deendayal Energy University, India. E-mail: rutvij.jhaveri@sot.pdpu.ac.in

Sagar V. Ramani is with Dept. of Computer Engr., Gujarat Technological University, India. E-mail: ramani_sagar@gtu.edu.in

Gautam Srivastava is with Dept. of Mathematics & Computer Science, Brandon University, Canada and also with Research Centre for Interneural Computing, China Medical University, Taichuing, Taiwan. E-mail: srivastavag@brandonu.ca

Thippa Reddy Gadekallu is with School of Information Technology and Engr., Vellore Institute of Technology, India. E-mail: thippareddy.g@vit.ac.in

Vaneet Aggarwal is with School of IE and ECE, Purdue University, USA. E-mail: vaneet@purdue.edu

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3104499, IEEE Transactions on Network Science and Engineering

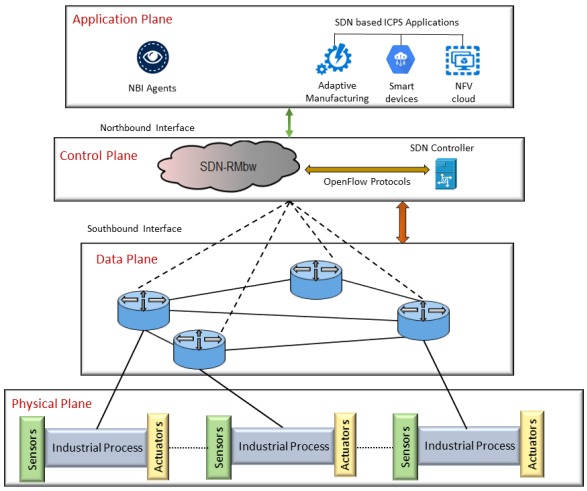

Fig. 1: Software-defined networking architecture for industrial cyber-physical systems

implemented on the top of a Ryu SDN controller[1] and its performance is evaluated on a hardware testbed in terms of success rate, network throughput and path restoration delay. Moreover, to test the scalability of the proposed framework, we use the Mininet tool to carry out emulations on different network topologies. The experimental results prove the superiority of the proposed SDN-based framework when compared with an existing approach along with other baseline approaches and, provides improved fault-resilience to the aforementioned events. The proposed framework is aligned with self-reconfigurability and self-optimization, which are the objectives of Industry 4.0 as discussed in [4].

The rest of the paper is organized as follows: Section II reviews the related work. The proposed SDN-based framework is discussed in Section III. The experimental setup and results are discussed in Section IV. Section V concludes the proposed work.

II. Related Work

This section reviews the state-of-the-art SDN-based fault-resilient approaches. In [2], a fault recovery mechanism is presented which aims at recovery from a link failure. The connectivity of each link is constantly monitored during normal mode. When a link failure is detected, the system shifts to an emergency mode where each switch is assumed to have backup rules in its shadow table. The mechanism, thus, aims at fault recovery within minimal and bounded delays. Meanwhile, it is to be noted that the mechanism assumes each switch to have backup rules, and also, it requires constant monitoring of links which adds to communication overhead in the network. In [5], an SDN-based framework is proposed aiming at fault recovery within a limited time. The proposed framework attempts to optimize the path restoration delay to satisfy the fault tolerance constraints of firm real-time flows. The work demonstrates the optimization of path restoration delay by minimizing the delays incurred during path recalculation, path reassignment, and fault recognition phases. However, the framework does not have support for fault recovery during multiple-link failures. In [6], an SDN-based failure recovery approach is proposed for smart grid networks with real-time data monitoring for improving system reliability and for managing demand and resources. With the assumption that the source point receives acknowledgment messages for data forwarding, the proposed mechanism can detect failures using interruption messages received from the source point. However, considering this assumption, the system may generate significant communication overhead. Furthermore, the mechanism does not consider the distinct requirements of multiple traffic flows. In [7], a cyber-security and resilience framework for SDN-based industrial manufacturing system is proposed. Based on the disruption caused by an attack, the equilibrium resilience mechanisms recompute a new secure network path and attempt to recover to the required security state. It is to be noted that the proposed resilience framework has not been implemented. Moreover, it does not take network failures into account.

In [8], a communication architecture based on SDN is proposed for microgrids. This architecture aims at providing resilience to both, cyber as well as physical faults. The proposed mechanism, using SDN-controller, provides automatic failover, traffic prioritization, and delay guarantees to deal with various network issues such as bandwidth allocation, port failure, and data congestion. However, the proposed architecture does not consider the distinct requirements of multiple traffic flows. To mitigate this type of multiple traffic flows in a heterogeneous environment, network resiliency becomes a key challenge [9], [10]. To achieve such resiliency in a dense heterogeneous network, artificial intelligence (AI) and machine learning (ML) can play a vital role in handling real-time big data in dynamic environments [11], [12]. The authors in [13] discuss the security breaches while applying machine learning in resilience approaches proposed in [14], [15], [16]. In [17], a communication network based on SDN architecture is proposed to improve the resilience and security in operations of a microgrid. The proposed mechanism hypothetically detects scenarios such as link failure, resource utilization, and delay requirements. Based on this periodic review, the SDN controller optimizes the failover paths. In the second stage, the mechanism detects a cyber attack during run-time and carries on reconfiguration if a better solution is found based on the current network condition. However, the proposed mechanism does not consider the distinct requirements of multiple traffic flows. In [18], a two-stage fault resilience mechanism is proposed to address the issue of link failures in SDN. To achieve fast and customized recovery from a fault, the mechanism takes into account: (1) Storage resource consumption (2) recovery time, and (3) quality-of-service (QoS). The first stage attempts to guarantee the fault recovery within an acceptable time using virtual LAN features and fast failover in OpenFlow protocol. In the second stage, memory is optimized and, the network resources are reallocated based on the bandwidth and delay needs. However, as a switch may have limited memory, the problem of storing multiple backup paths may be raised due to the fast failover mechanism in this limited memory. Besides, the mechanism does not address

[1] https://ryu-sdn.org/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3104499, IEEE Transactions on Network Science and Engineering

the challenge of managing bandwidth requirements or delay requirements of traffic flows. The authors in [19] propose a multipath resilient routing system using SDN called MPResiSDN, which is a reactive resilient mechanism for providing connectivity in smart city networks in the presence of failures. The proposed system comprises six different components. The system requires fewer paths for a relatively trivial condition, while it requires more number of paths for a more severe challenge. However, the system induces higher overhead as the number of paths increases. The emulation results with a Ryu controller depict that the system provides improved performance than a baseline mechanism.

Each of the aforementioned works either is not evaluated with hardware testbed or has inherent limitations such as lack of *self-reconfigurability* and *self-optimization*. Self-reconfigurability is imperative in the case of single/multiple link failures in the network. Meanwhile, when network requirements change, the need for QoS parameters also changes dynamically. To satisfy these dynamic changes in requirements, there is a need for *self-optimization* in the network. Therefore, the proposed approach attempts to overcome the challenges of *self-reconfigurability* and *self-optimization*. To accommodate the dynamic changing requirement of traffic flows, we devise a contract-based fault-resilience scheme that attempts to manage bandwidth requirements of multiple flows: (1) when the flows change their bandwidth requirements at run-time and, (2) when communication gets disrupted by link failures. It is to be noted that the mechanism proposed in [5] complements our proposed mechanism and therefore, it can be integrated with our mechanism.

III. Proposed SDN-based Fault-Resilience Framework

In our previous work [4], we proposed an SDN-based framework to manage industrial communication delays. Complementing our previous work [4], we propose an SDN-based framework for managing bandwidth in industrial networks. The proposed framework, termed as Software-Defined Networking Resilience Management for Bandwidth (SDN-RMbw), resides on the top of the SDN controller and adapts a contract-based methodology to manage bandwidth requirements of distinct traffic flows. SDN-RMbw attempts to achieve zero downtime for traffic flows by providing fault resilience to events such as dynamic changes in bandwidth requirements of the traffic flows and link failures. SDN-RMbw possesses a hybrid strategy (proactive and reactive): (1) with the proactive strategy, it periodically observes the current network state and takes suitable actions, (2) with the reactive strategy, it reacts to a fault immediately to provide fault-resilient communication.

A. Framework Components

Fig. 2 shows five different components of the proposed framework, SDN-RMbw.

(1) **Bandwidth Contracts:** A contract [20] is a robust technique to rapidly detect faults in a system. In this framework, a bandwidth contract is proposed to define the bandwidth guarantees between a source-destination pair by specifying

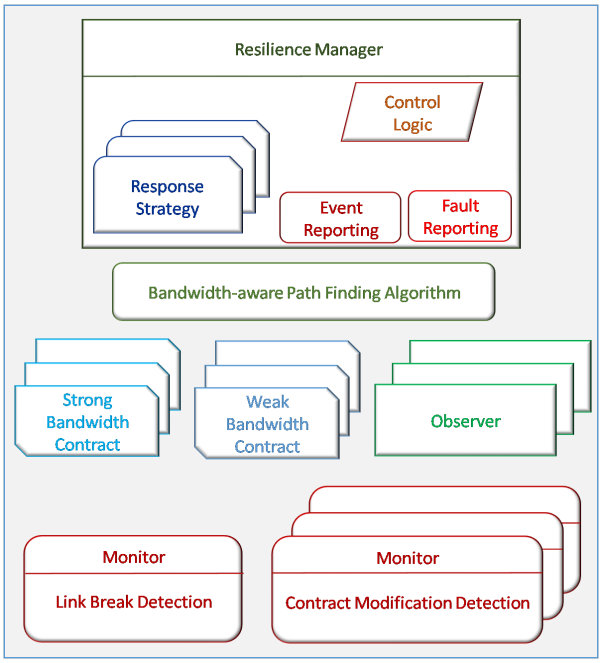

Fig. 2: Components of the proposed SDN-based fault-resilience framework

inputs, outputs, parameters, assumptions, and guarantees. Fig. 3 depicts an example of a bandwidth contract between a source, si and a destination, sj. The bandwidth contract describes the guarantee of the estimated minimum available bandwidth termed as EBW_{si-sj} to be greater than or equal to the bandwidth requirement of the flow from si to sj, termed as RBW_{si-sj}. The estimation of the minimum available bandwidth is carried out during each of the estimation intervals. In this work, we propose two types of bandwidth contracts: (i) strong contracts specifying the bandwidth requirements, (ii) weak contracts specifying the tolerable bandwidth requirements in case of violation of strong contracts. We consider the manual generation of both types of contracts based on the given bandwidth requirements. It is to be noted that we consider dynamic updates of both types of contracts.

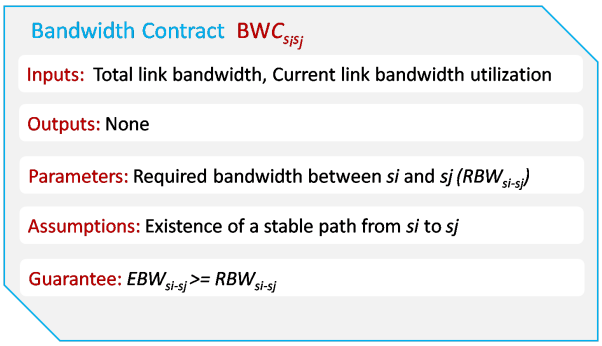

Fig. 3: Example bandwidth contract

(2) **Observers:** An observer detects any violation in the concerned bandwidth contract. The violation of a contract is known as a fault in the system. It reports a fault to the resilience manager.

(3) **Monitors:** A monitor is responsible to detect the events which may cause a fault (contract failure). In this work, we consider two events that may induce faults in the system: ($E1$) run-time changes in the bandwidth requirements, and ($E2$) link failure. The monitor for detecting event $E1$ (termed as a contract modification monitor) continuously observes any updates in the corresponding bandwidth contract, while the monitor for detecting event $E2$ (termed as link failure monitor) utilizes the received port statistics of switches. A monitor reports occurrence of an event to the resilience manager.

(4) **Resilience Manager:** The resilience manager (RM) is the key component of the proposed framework which is responsible for providing fault resilience in the network. After receiving the reported fault, it executes one of the following response strategies with the help of control logic: *(RS1)* reassign paths, *(RS2)* switch to the corresponding weak contract and reassign paths or, *(RS3)* issue an alert.

(5) **Bandwidth-aware Path-finding Algorithm:** The bandwidth-aware path-finding algorithm calculates the data forwarding path between the desired source-destination pair. The algorithm adapts *Dijkstra's shortest path algorithm* to find an end-to-end path containing maximum available bandwidth between the source-destination pair.

B. Interaction among Framework Components

Fig. 4 shows the interaction among the different components of SDN-RMbw. When any of the two events $E1$ or $E2$ is detected by the corresponding monitor, it reports it to the RM. The RM immediately triggers the bandwidth-aware path-finding algorithm to find a new path containing maximum available bandwidth between the source-destination pair at the current point of time. After receiving the value of maximum available bandwidth, the RM triggers the observer to check whether the newly discovered path satisfies the bandwidth requirements specified in the strong contract. If so, the revised path is reassigned to the switches. However, if the RM receives the report of a fault from an observer due to a strong contract failure, the RM asks the observer to check the bandwidth requirements specified in the weak contract. The RM switches to the weak contract and reassigns the revised path if the bandwidth requirements of the weak contract are satisfied. In the worst case, when the fault is reported due to the failure of a weak contract, the RM issues an alert message so that the responsible personnel may take the necessary action.

IV. Experimental Results and Performance Evaluation

In the experiments, we consider wired networks. The performance of SDN-RMbw is evaluated by:

- **Success rate:** It gives the measure of resilience by taking into consideration the rate of bandwidth requirements satisfaction.
- **Network throughput:** It gives the measure of the rate of data delivery. When the success rate is higher, the network throughput is also higher.
- **Path restoration delay:** It gives the measure of total delay incurred during path reassignment, path recalculation, and fault detection.

The performance of SDN-RMbw is compared with the following three baseline mechanisms: (1) **SDN-woRMbw**: SDN controller without any resilience management, (2) **SDN-sRMbw**: resilience management with only strong bandwidth contracts and, (3) **SDN-pRMbw**: resilience management with the only proactive strategy where fault detection takes place periodically during the estimation interval. Table I summarizes the characteristic comparison of the four mechanisms.

TABLE I: Characteristic Comparison between Mechanisms

Mechanism	Proactive Strategy	Reactive Strategy	Strong Bandwidth Contracts	Weak Bandwidth Contracts
SDN-woRMbw	-	-	-	-
SDN-sRMbw	✓	✓	✓	-
SDN-pRMbw	✓	-	✓	✓
SDN-RMbw	✓	✓	✓	✓

We consider the estimation interval of 10 seconds in the experiments, as we assume a stable network state within this period. We conduct various tests by varying: (1) bandwidth utilization and, (2) number of events $E1$ and/or $E2$.

A. Hardware Testbed Results

The experiments are carried out on a hardware testbed to evaluate the performance of SDN-RMbw as shown in Fig. 5. The configuration of hardware and software for a controller and that of other hardware are shown in Table II. Two hosts are connected to Port 4 of two distinct Zodiac Fx switches where the Zodiac Fx switches are connected in a triangular topology. The Zodiac Fx switches are connected to the system hosting a Ryu SDN controller and SDN-RMbw application. iPerf tool is used to generate TCP traffic flows in the network. Each flow of 20 Mb is sent with a time interval of 1 second on a 100 Mbps Ethernet link.

TABLE II: Hardware and Software Configuration

Hardware/Software	Specification
Host CPU	Intel Core - i7 8550U @1.99 GHz
Host Memory	8 GB
Host Operating System	Windows 10
Virtualization	VirtualBox 6.0
Guest OS	Ubuntu 18.04
VM Configuration	2 CPU Cores, 4 GB Memory
Ryu SDN Controller	Version 4.26
ZodiacFx OpenFlow Switch	Firmware 0.86
Hub	DAX Networks- DX-924U-ARO

1) Varying Bandwidth Utilization: In this set of experiments, the bandwidth utilization is varied from 20% to 100% for the source-destination pair.

*a) **Test 1: Regarding link failure**:* In this test, an event ($E2$) is generated to induce a fault during the experiment. During the experiment, the link between the two switches connecting hosts is broken once.

The success rate of all the mechanisms decreases as the bandwidth utilization increases as shown in Fig. 6a. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3104499, IEEE Transactions on Network Science and Engineering

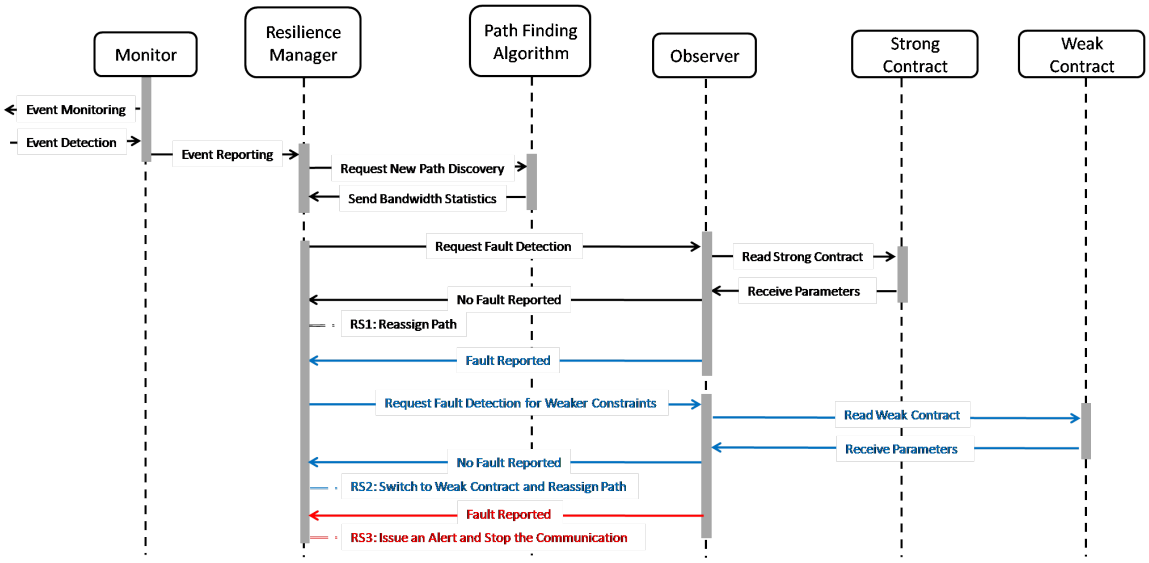

Fig. 4: Interaction among components of SDN-RMbw

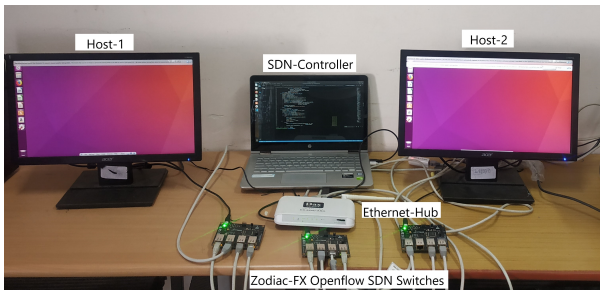

Fig. 5: Example bandwidth contract

reason behind this is, as the bandwidth utilization increases, the congestion in the network also increases. The increase in congestion leads to more number of faults and therefore, a lower success rate. As SDN-woRMbw does not have any resilience strategy, it provides the highest faults, and therefore it has the lowest average success rate of 76.89%. SDN-sRMbw provides a higher average success rate of 81.89% as it contains a resilience mechanism with strong contracts. On the other hand, as SDN-pRMbw has a provision of weak contracts which state the worst-case bandwidth requirements of the flows, it provides a 3.98% improvement in the success rate than SDN-sRMbw. This suggests that it is imperative to plan the worst-case bandwidth requirements for ICPS which are prone to faults. Meanwhile, as SDN-RMbw possesses a hybrid strategy (proactive and reactive) along with the planning of worst-case bandwidth requirements using weak contracts, it induces less number of faults and therefore, it provides the highest average success rate of 86.99%. The results state that improved network resilience is provided by SDN-RMbw as compared to the baseline mechanisms.

As depicted in Fig. 6b, as the bandwidth utilization increases, the average network throughput increases as the rate of packets reaching the destination increases. SDN-woRMbw provides the lowest average network throughput of 53.98 Mbps as it does not contain any resilience mechanism and induces more faults in the network. On the other hand, SDN-sRMbw and SDN-pRMbw provide an average network throughput of 54.79 Mbps and 55.75 Mbps respectively due to the aforementioned reasons. Moreover, SDN-RMbw improves the QoS by providing the highest average network throughput of 57.98 Mbps as it induces the least faults.

As SDN-pRMbw does not possess any reactive strategy, it gives the average path restoration delay of 10.186 seconds. Meanwhile, the average path restoration delays for SDN-sRMbw and SDN-RMbw are 0.383 ms and 0.348 ms respectively.

b) ***Test 2: Regarding run-time changes in bandwidth requirements****:* In this set of tests, different number of events $E1$ are triggered to generate a different number of faults during the experiments.

Test 2 (A)*: Triggering event* $E1$ *one time:* In this test, fault generating event ($E1$) is triggered once during the experiment. Fig. 6c and Fig. 6d present average success rates and average network throughput respectively. The results of Fig. 6c and Fig. 6d along with path restoration delays are summarized in Table III.

Test 2 (B)*: Triggering event* $E1$ *five times:* In this test, fault generating events ($E1$) are triggered five times at different time instances during the experiment. Fig. 6e and Fig. 6f present average success rates and average network throughput respectively. The results of Fig. 6e and Fig. 6f along with path restoration delays are summarized in Table IV.

c) ***Test 3: Regarding link failure and run-time changes in delay requirements****:* In this test, events ($E1$) are generated four times and event ($E2$) is generated once at different time instances. Fig. 7a and Fig. 7b present average success rates and average network throughput respectively. The results of Fig. 7a and Fig. 7b are summarized in Table V.

Observations: The above tests depict that the success rate decreases eventually when bandwidth utilization increases. This is due to network congestion which causes more faults in the network by not satisfying the requirements of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3104499, IEEE Transactions on Network Science and Engineering

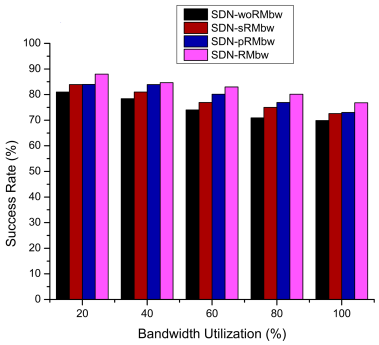

Fig. 6a: Success rate during a link-break ($E2$) with varying bandwidth utilization

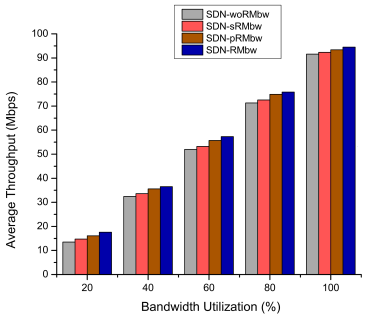

Fig. 6b: Throughput during a link-break ($E2$) with varying bandwidth utilization

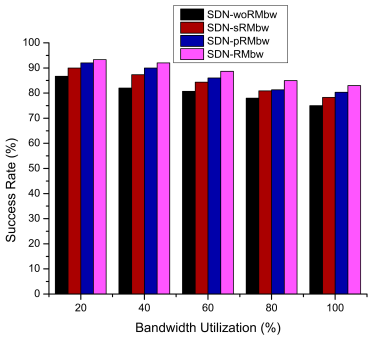

Fig. 6c: Success rate during 1 change in bandwidth requirement ($E1$) with varying bandwidth utilization

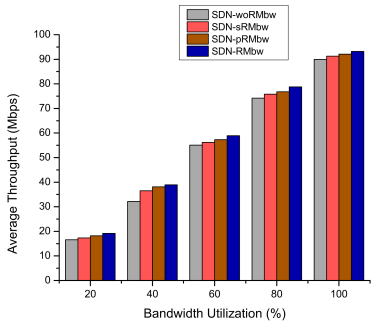

Fig. 6d: Throughput during 1 change in bandwidth requirement ($E1$) with varying bandwidth utilization

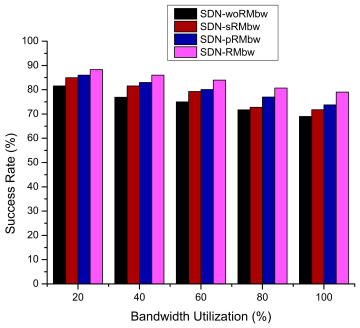

Fig. 6e: Success rate during 5 changes in bandwidth requirement ($E1$) with varying bandwidth utilization

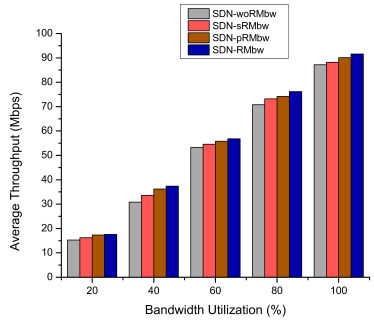

Fig. 6f: Throughput during 5 changes in bandwidth requirement ($E1$) with varying bandwidth utilization

Fig. 6: Evaluation Results for Linkbreak ($E1$) and Bandwidth Requirement ($E2$)

TABLE III: Result summary of Test 2 (A)

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	80.47 %	54.43 Mbps	-
SDN-sRMbw	82.17 %	55.31 Mbps	0.189 ms
SDN-pRMbw	85.93 %	56.74 Mbps	10.137 seconds
SDN-RMbw	88.40 %	58.95 Mbps	0.173 ms

TABLE IV: Result summary of Test 2 (B)

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	75.78 %	51.43 Mbps	-
SDN-sRMbw	81.36 %	53.31 Mbps	0.428 ms
SDN-pRMbw	84.10 %	54.74 Mbps	10.383 seconds
SDN-RMbw	87.10 %	55.95 Mbps	0.419 ms

flows. Moreover, SDN-RMbw provides the highest success rate and highest network throughput and therefore, it possesses significant fault-resilience capability as compared to the other baseline mechanisms.

2) Varying Number of Events: The numbers of events ($E1$ and/or $E2$) are varied at bandwidth utilization of 60% in this set of tests.

a) ***Test 4: Regarding run-time changes in bandwidth requirements****:* The bandwidth requirements are changed ($E1$) from one to five times at different time instances in this test. It is obvious that the number of faults increases as the number of events increases. As a result, both the success rate and throughput of all the mechanisms decrease with an increasing number of events as shown in Fig. 7c and Fig. 7d. The results of Fig. 7c and Fig. 7d along with path restoration delays are depicted in Table VI.

b) ***Test 5: Regarding link failure and run-time changes in bandwidth requirements****:* In this test, events ($E1$ and $E2$) are generated different number of times at different time instances. Fig. 7e and Fig. 7f present average success rates and average network throughput respectively. The results of Fig. 7e and Fig. 7f are depicted in Table VII.

TABLE V: Result summary of Test 3

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	73.97 %	50.68 Mbps	-
SDN-sRMbw	78.59 %	52.67 Mbps	0.428 ms
SDN-pRMbw	81.86 %	53.49 Mbps	10.342 seconds
SDN-RMbw	85.32 %	55.02 Mbps	0.423 ms

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3104499, IEEE Transactions on Network Science and Engineering

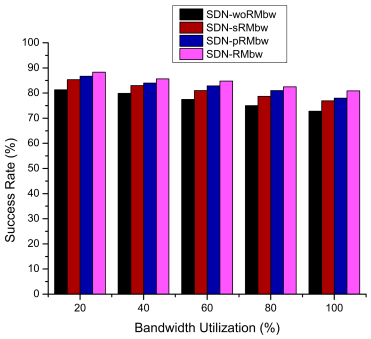

Fig. 7a: Success rate during 5 events ($E1$ and $E2$) with varying bandwidth utilization

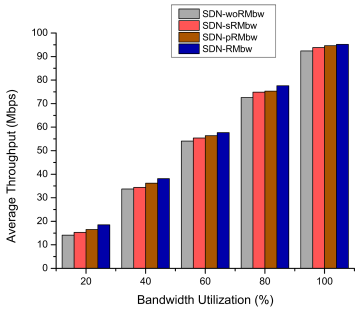

Fig. 7b: Throughput during 5 events ($E1$ and $E2$) with varying bandwidth utilization

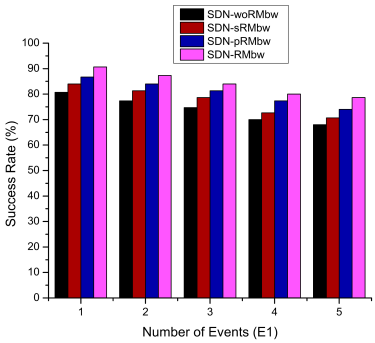

Fig. 7c: Success rate with varying number of events ($E1$)

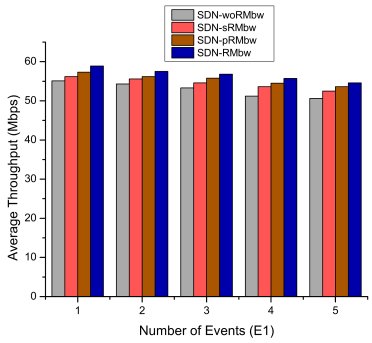

Fig. 7d: Throughput with varying number of events ($E1$)

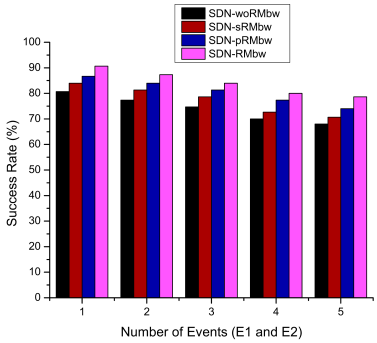

Fig. 7e: Success rate with varying number of events ($E1$ and $E2$)

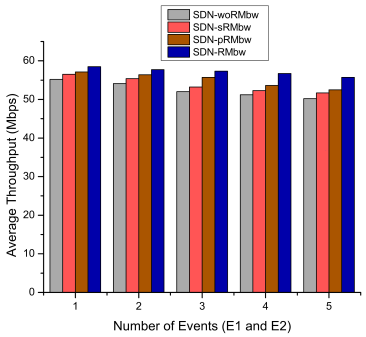

Fig. 7f: Throughput with varying number of events ($E1$ and $E2$)

Fig. 7: Evaluation Results with Varying Number of Events

TABLE VI: Result summary of Test 4

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	74.40 %	52.90 Mbps	-
SDN-sRMbw	80.93 %	54.50 Mbps	0.452 ms
SDN-pRMbw	84 %	55.58 Mbps	10.271 seconds
SDN-RMbw	87.87 %	56.70 Mbps	0.340 ms

TABLE VII: Result summary of Test 5

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	74.13 %	52.54 Mbps	-
SDN-sRMbw	77.47 %	53.82 Mbps	0.427 ms
SDN-pRMbw	80.67 %	54.60 Mbps	10.313 seconds
SDN-RMbw	84.13 %	57.17 Mbps	0.425 ms

Observations: The above tests depict that SDN-RMbw provides the highest throughput and success rate with notable path restoration delay. The results of the success rate depict that SDN-RMbw provides significant fault-resilience capability as compared to the other baselines mechanisms.

B. Emulation Results

In this set of experiments, the scalability of SDN-RMbw is evaluated on the Mininet emulator using a Ryu SDN controller. Emulations are carried out on a 20-switch mesh network as shown in Fig. 10. Table VIII summarizes the experimental parameters. Each flow of 200 Mb is sent with a time interval of 1 second on a 1 Gbps Ethernet links.

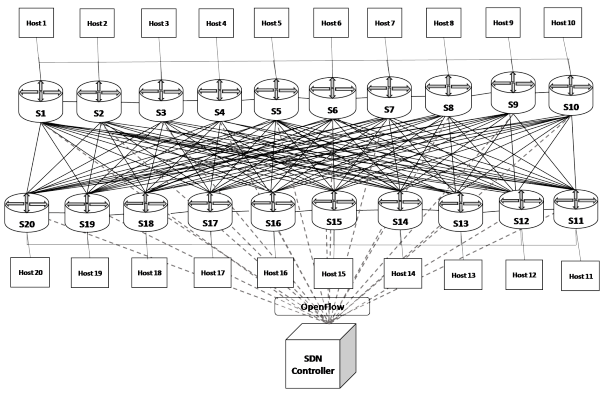

Fig. 10: Mesh network with 20 OpenFlow switches

1) Varying Bandwidth Utilization: In this set of tests, the bandwidth utilization is varied from 20% to 100% by generating events ($E1$ and/or $E2$) five times.

a) ***Test 6: Regarding link failure****:* In this test, events ($E2$) are generated five times in order to induce different number of faults during the experiments. Fig. 8a and Fig. 8b

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3104499, IEEE Transactions on Network Science and Engineering

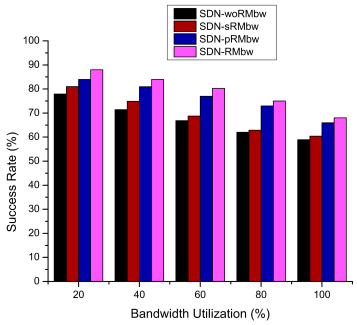

Fig. 8a: Success rate during 5 link-breaks ($E2$) with varying bandwidth utilization

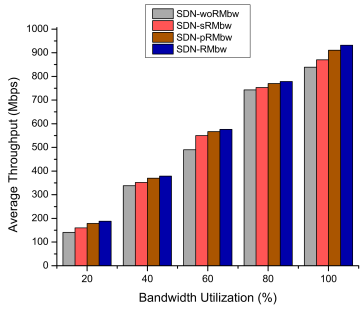

Fig. 8b: Throughput during 5 link-breaks ($E2$) with varying bandwidth utilization

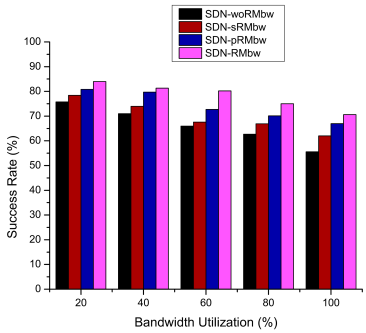

Fig. 8c: Success rate during 5 changes in bandwidth requirements ($E1$) with varying bandwidth utilization

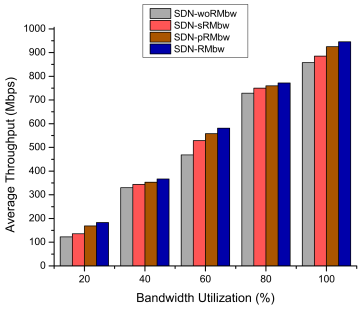

Fig. 8d: Throughput during 5 changes in bandwidth requirements ($E1$) with varying bandwidth utilization

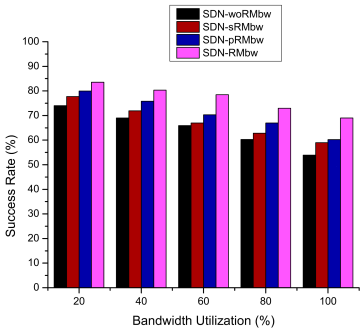

Fig. 8e: Success rate during 5 events ($E1$ and $E2$) with varying bandwidth utilization

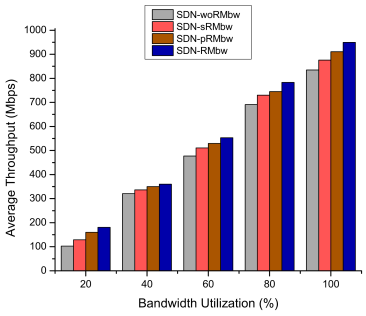

Fig. 8f: Throughput during 5 events ($E1$ and $E2$) with varying bandwidth utilization

Fig. 8: Evaluation Results during during 5 Linkbreaks ($E2$) and Changes in Bandwidth Requirements

TABLE VIII: Emulation Parameters

Parameters	Value
Link capacity	1 Gbps
Bandwidth utilization (Varying)	20 % to 100%
Emulation time	300 seconds
Number of Events $E1$ and/or $E2$ (Varying)	1 to 5
Traffic type	TCP
Traffic generation	iPerf
Number of hosts	20
Number of source-destination pairs	4

present average success rates and average network throughput respectively. The results of Fig. 8a and Fig. 8b are depicted in Table IX.

*b) **Test 7: Regarding run-time changes in bandwidth requirements**:* In this test, events ($E1$) are generated five times in order to induce different number of faults during the experiments. Fig. 8c and Fig. 8d present average success rates and average network throughput respectively. The results of Fig. 8c and Fig. 8d are depicted in Table X.

*c) **Test 8: Regarding link failure and run-time changes in bandwidth requirements**:* In this test, events ($E1$) are generated three times and events ($E2$) are generated two times at different instances of time during the experiments. Fig. 8e and Fig. 8f present average success rates and average network throughput respectively. The results of Fig. 8e and Fig. 8f are depicted in Table XI.

TABLE IX: Result summary of Test 6

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	67.40 %	510.11 Mbps	-
SDN-sRMbw	69.58 %	536.92 Mbps	0.369 ms
SDN-pRMbw	76.16 %	559.26 Mbps	10.195 seconds
SDN-RMbw	79.04 %	570.68 Mbps	0.249 ms

TABLE X: Result summary of Test 7

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	66.19 %	501.55 Mbps	-
SDN-sRMbw	69.76 %	528.72 Mbps	0.325 ms
SDN-pRMbw	74.06 %	553.03 Mbps	10.530 seconds
SDN-RMbw	78.21 %	569.67 Mbps	0.286 ms

2) Varying Number of Events: The number of events ($E1$ for the whole network, while $E2$ for individual source-destination pair) are varied from 1 to 5 at bandwidth utilization of 60% to generate multiple faults in this set of tests.

TABLE XI: Result summary of Test 8

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	64.61 %	485.24 Mbps	-
SDN-sRMbw	67.47 %	516.37 Mbps	0.343 ms
SDN-pRMbw	70.65 %	539.08 Mbps	10.341 seconds
SDN-RMbw	76.85 %	565.11 Mbps	0.322 ms

a) ***Test 9: Regarding link failure****:* In this test, the link is broken ($E2$) between distinct switches in the network different number of times. Fig. 9a and Fig. 9b present average success rates and average network throughput respectively. The results of Fig. 9a and Fig. 9b are depicted in Table XII.

b) ***Test 10: Regarding run-time changes in bandwidth requirements****:* In this test, the bandwidth requirements are changed ($E1$) different number of times at different time instances. Fig. 9c and Fig. 9d present average success rates and average network throughput respectively. The results of Fig. 9c and Fig. 9d are depicted in Table XIII.

c) ***Test 11: Regarding link failure and run-time changes in bandwidth requirements****:* In this test, events ($E1$ and $E2$) are generated different number of times at different time instances. Fig. 9e and Fig. 9f present average success rates and average network throughput respectively. The results of Fig. 9e and Fig. 9f are depicted in Table XIV.

TABLE XII: Result summary of Test 9

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	66.92 %	472.91 Mbps	-
SDN-sRMbw	73.99 %	522.99 Mbps	0.394 ms
SDN-pRMbw	76.82 %	548.99 Mbps	10.381 seconds
SDN-RMbw	81.93 %	561.99 Mbps	0.329 ms

TABLE XIII: Result summary of Test 10

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	64.98 %	467.36 Mbps	-
SDN-sRMbw	75.79 %	570.96 Mbps	0.310 ms
SDN-pRMbw	79.71 %	573.16 Mbps	10.249 seconds
SDN-RMbw	85.89 %	578.92 Mbps	0.288 ms

TABLE XIV: Result summary of Test 11

Mechanism	Success Rate	Throughput	Path Restoration Delay
SDN-woRMbw	65.68 %	469.84 Mbps	-
SDN-sRMbw	72.79 %	521.59 Mbps	0.389 ms
SDN-pRMbw	75.39 %	539.65 Mbps	10.339 seconds
SDN-RMbw	82.99 %	568.10 Mbps	0.347 ms

Observations: The emulation results are consistent with the observations obtained from the hardware testbed as depicted from the above tests. This shows that SDN-RMbw is scalable.

C. Result comparison of SDN-RMbw with MPResiSDN

For any mechanism, the throughput of the network shows the actual rate of the traffic received at the destination side, and therefore, throughput is one of the important QoS metrics. Therefore, in this test, we compare the throughput of our approach, SDN-RMbw, with an existing approach, MPResiSDN [19]. In Fig. 10, we present the results for the network throughput with varying k values for SDN-RMbw and MPResiSDN, where k is the number of established paths.

Observations: The test depicts that for both the approaches as k values increase network throughput also increase. This means that as k values increase, multiple paths help to deliver packets in parallel to their destination which provides higher throughput. Table XV for different k values suggest that average network throughput of SDN-RMbw is always higher as compared to MPResiSDN as SDN-RMbw possesses hybrid mechanism and provision for worst-case bandwidth requirements.

TABLE XV: Result summary of Average Network Throughput of SDN-RMbw and MPResiSDN

k values	Average Network Throughput of SDN-RMbw	Average Network Throughput of MPResiSDN
1	1.835 Mbps	1.042 Mbps
2	2.713 Mbps	2.024 Mbps
3	3.91 Mbps	3.013 Mbps

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2021.3104499, IEEE Transactions on Network Science and Engineering

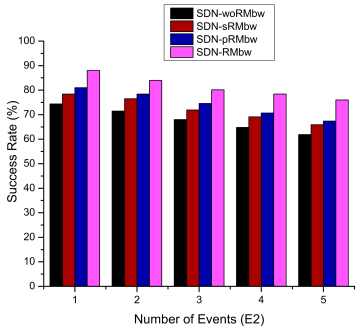

Fig. 9a: Success rate with varying number of events ($E2$)

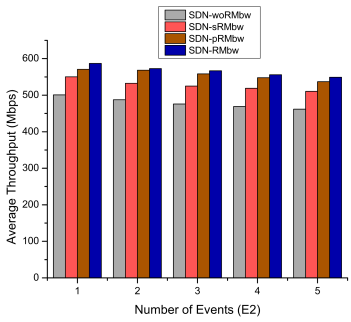

Fig. 9b: Throughput with varying number of events ($E2$)

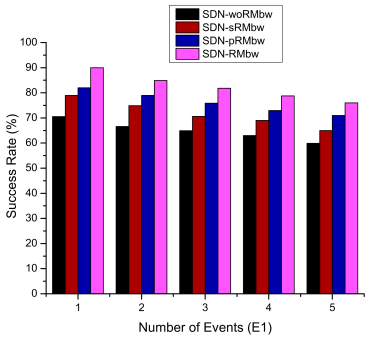

Fig. 9c: Success rate with varying number of events ($E1$)

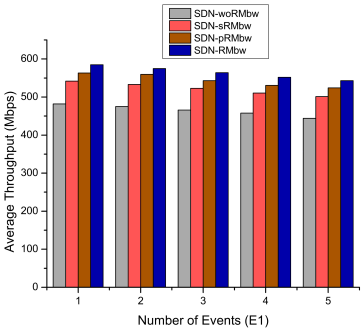

Fig. 9d: Throughput with varying number of events ($E1$)

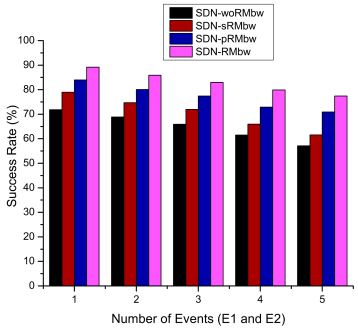

Fig. 9e: Success rate with varying number of events ($E1$ and $E2$)

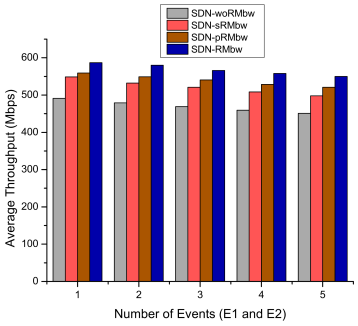

Fig. 9f: Throughput with varying number of events ($E1$ and $E2$)

Fig. 9: Evaluation Results Varying Number of Events

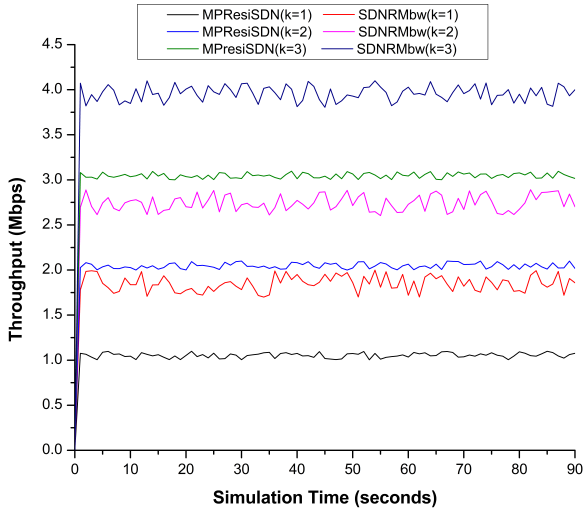

Fig. 10: Comparison of network throughput of SDN-RMbw with MPResiSDN for varying k values

V. Conclusion

Communication in an IPCS typically demands fault-tolerance for achieving bandwidth guarantees. It becomes challenging to provide fault-resilience in such systems which are prone to link failures and where bandwidth requirements change dynamically. To address these challenges, an SDN-based fault-resilient framework is presented in this paper for wired networks which uses a contract-based methodology to manage dynamic bandwidth requirements of multiple traffic flows. The framework defines bandwidth contracts to state bandwidth guarantees of traffic flows. Monitors are used to detecting events that may cause faults while observers are used to detect faults. The proposed framework is evaluated using metrics such as success rate, network throughput, and path restoration delay. The experiments were carried out on a hardware testbed with a Ryu SDN controller depict that the proposed framework provides the highest success rate compared to the baseline mechanisms. Thus, the proposed framework provides a significant level of fault resilience. Furthermore, the proposed framework improves quality-of-service by providing the highest network throughput and, notable path restoration delay during fault recovery. Moreover, the emulation results on a 20-switch mesh network demonstrate that the proposed framework is scalable. Consequently, the results suggest that a resilience manager operating proactively and reactively provides considerable improvement in fault-resilience capability than a proactive-only mechanism. Besides, the results suggest that it is imperative to plan worst-case bandwidth requirements for critical industrial applications. However, the proposed framework can be incorporated

with ML techniques to provide more reliable fault-tolerant operations in SDN. Moreover, AI/ML can prove to be vital components in the management and orchestration of such a dynamic network.

ACKNOWLEDGMENTS

The authors would like to thank Professor Rui Tan and Professor Arvind Easwaran of NTU, Singapore for their guidance and motivation. Special thanks to Paul Zanna, the founder, and CEO of Northbound Networks, Australia, for his valuable input. Gautam Srivastava's work is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] S. Meraghni, L. S. Terrissa, S. Ayad, N. Zerhouni, and C. Varnier, "Post-prognostics decision based on cyber-physical systems," 05 2017.

[2] K. Lee, M. Kim, T. Park, H. S. Chwa, J. Lee, S. Shin, and I. Shin, "Mc-sdn: Supporting mixed-criticality real-time communication using software-defined networking," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6325–6344, 2019.

[3] L. Esterle and R. Grosu, "Cyber-physical systems: challenge of the 21st century," *e & i Elektrotechnik und Informationstechnik*, vol. 133, no. 7, pp. 299–303, 2016.

[4] R. H. Jhaveri, R. Tan, A. Easwaran, and S. V. Ramani, "Managing industrial communication delays with software-defined networking," in *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2019, pp. 1–11.

[5] K. Lee, M. Kim, H. Kim, H. S. Chwa, J. Lee, and I. Shin, "Fault-resilient real-time communication using software-defined networking," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2019, pp. 204–215.

[6] S. Al-Rubaye, E. Kadhum, Q. Ni, and A. Anpalagan, "Industrial internet of things driven by sdn platform for smart grid resiliency," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 267–277, 2017.

[7] R. F. Babiceanu and R. Seker, "Cyber resilience protection for industrial internet of things: A software-defined networking approach," *Computers in Industry*, vol. 104, pp. 47–58, 2019.

[8] L. Ren, Y. Qin, B. Wang, P. Zhang, P. B. Luh, and R. Jin, "Enabling resilient microgrid through programmable network," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2826–2836, 2016.

[9] J. Du, C. Jiang, H. Zhang, Y. Ren, and M. Guizani, "Auction design and analysis for sdn-based traffic offloading in hybrid satellite-terrestrial networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2202–2217, 2018.

[10] J. Du, E. Gelenbe, C. Jiang, H. Zhang, and Y. Ren, "Contract design for traffic offloading and resource allocation in heterogeneous ultra-dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2457–2467, 2017.

[11] J. Du, C. Jiang, J. Wang, Y. Ren, and M. Debbah, "Machine learning for 6g wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 122–134, 2020.

[12] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, R. Kaluri, D. S. Rajput, G. Srivastava, and T. Baker, "Analysis of dimensionality reduction techniques on big data," *IEEE Access*, vol. 8, pp. 54 776–54 788, 2020.

[13] R. Sagar, R. Jhaveri, and C. Borrego, "Applications in security and evasions in machine learning: A survey," *Electronics*, vol. 9, no. 1, p. 97, 2020.

[14] R. Macedo, J. Paulo, J. Pereira, and A. Bessani, "A survey and classification of software-defined storage systems," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–38, 2020.

[15] A. Hussein, A. Chehab, A. Kayssi, and I. H. Elhajj, "Machine learning for network resilience: The start of a journey," in *2018 Fifth International Conference on Software Defined Systems (SDS)*. IEEE, 2018, pp. 59–66.

[16] S. Gangadhar and J. P. Sterbenz, "Machine learning aided traffic tolerance to improve resilience for software defined networks," in *2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM)*. IEEE, 2017, pp. 1–7.

[17] D. Jin, Z. Li, C. Hannon, C. Chen, J. Wang, M. Shahidehpour, and C. W. Lee, "Toward a cyber resilient and secure microgrid using software-defined networking," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2494–2504, 2017.

[18] L. Wang, L. Yao, Z. Xu, G. Wu, and M. S. Obaidat, "Cfr: A cooperative link failure recovery scheme in software-defined networks," *International Journal of Communication Systems*, vol. 31, no. 10, p. e3560, 2018.

[19] S. L. Aljohani and M. J. Alenazi, "Mpresisdn: Multipath resilient routing scheme for sdn-enabled smart cities networks," *Applied Sciences*, vol. 11, no. 4, p. 1900, 2021.

[20] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. G. Larsen, "Contracts for systems design: Theory," Ph.D. dissertation, Inria Rennes Bretagne Atlantique; INRIA, 2015.