

SD-FFR: Software Defined Fast Failure Recovery Mechanism in the Automatic Warehouse

Ting-Cian Bai and Chin-Ya Huang

Abstract—Due to the rapid development of IoT technology, automatic guided vehicles (AGVs) interact with an industrial control system (ICS) through the wireless network to support the freight distribution in the automated warehouse. However, the message exchange among AGVs and the ICS would experience large packet loss or long transmission delay, in the presence of wireless network link failure and link congestion. Therefore, the performance of warehouse automation would be degraded. In this paper, we propose the Software Defined Fast Failure Recovery (SD-FFR) mechanism, aiming to improve network reliability and avoid link congestion caused by load unbalance. With SD-FFR, when link failure occurs, the SDN controller actively detects the link failure within milliseconds, and then reroutes the affected traffic flows accordingly. The proposed SD-FFR mechanism also takes load balance into consideration in selecting routing paths when a new traffic flow joins or link failure occurs in the network. The evaluation results show that network performance can be time efficiently improved.

Index Terms—SDN, Fast Failure Recovery, Load Balance, Wireless Mesh Network.

I. INTRODUCTION

To support large number of mobile devices working in the automatic warehouse, WiFi is considered to provide network services [1] because of its low cost comparing to 4G/5G. In this way, the automatic guided vehicles (AGVs) in the automatic warehouse can easily communicate with the industrial control system (ICS) to exchange maintenance messages, even when they are on the move. For example, a wireless network architecture, Industrial Automation-Factory Automation (WIA-FA), is designed and implemented in factory automation [1]. Furthermore, some researches also plan differently to react to different AGV usage scenarios in the industrial networks such as [2]. A flexible and reliable network is designed in the warehouse shuttle system by IEEE 802.15.4 to support dense goods placement in [2]. Authors in [3] combine the advantages of two network architecture, wireless mesh network and the enterprise network, to improve the performance of AGVs in transmitting unicast and multicast traffic. Specifically, the mesh capability is applied when applicable. Additionally, existing wireless communication technologies are investigated in satisfying the various latency and reliability requirements of data transmission to assist the use cases of AGV and the unmanned vehicle in industrial 4.0 [4]. Under this condition, the messages are exchanged through wireless links in the network. However, a link may be congested if too many messages are forwarded through it. On the other hand, a link may fail to

transmit message when fault occurs. When a link encounters congestion or failure, the message exchange may experience of loss or delay which in turn would introduce incorrect decision making and significantly degrade the warehouse operation. Specifically, to ensure the AGVs actually perform the required tasks in the automatic warehouse, the interruption or long delay in message exchange is not allowed [5].

To reliably and flexibly route traffic flows between the AGVs and the ICS, in this paper, we consider dual-band WiFi access points (APs) to form a network consisting of wired and wireless connectivity to route packets. Specifically, APs create a wireless mesh network (WMN) [6] by wirelessly connecting with each other. Further, the software defined network (SDN) [7] is applied in the network to dynamically route traffic flows according to the network status. SDN is a network architecture decoupling the network control plane and the data plane of a switch, and then a centralized SDN controller is introduced to control the packet forwarding in the whole network. In this way, the network management can be done globally. Considering the integration of SDN and WMN, authors in [8] adopt the OpenWrt framework on the development board aiming to build the real network environment of SDN integration with WMN architecture and determines the feasibility of SDN and WMN operation in the real network environment. In [9], [10], authors integrate WMN and SDN so that modularized and flexible routing strategies can be provided through SDN. In this way, the network performance can be enhanced because the strategies are designed to take advantages and eliminate the limitations of WMN by SDN. Wireless Mesh Software Defined Networks (wmSDN) combines the SDN and WMN aims to balance traffic loads on each wireless link which in turn enhances the user performance [11].

In SDN, two types of failure recovery strategies, active and passive strategy, are considered. On the other hand, the passive strategy reactively adjusts routing decisions when link failure occurs, while the active strategy pre-deploys backup resources in the network. For example, the shared ring method is proposed in [12] to reuse backup resources, aiming to cost effectively pre-deploying backup resources in TCAM. Although both recovery delay and the TCAM usage could be improved, the congestion may occurs in the post-recovery network. A particular method of the active recovery strategy is also developed in [13] by using RESTful API to per-forward the redundant rules periodically. Thus, the network services can be actively recovered, when the fault occurs. However, the backup rules are pushed periodically, therefore the failure may not be recovered immediately if the latest rules are not pushed. On the other hand, in [14], a reactive failure recovery

T.-C. Bai and C.-Y. Huang are with the Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taiwan. (e-mail: {M10802214, chinya}@gapps.ntust.edu.tw)

module in multi-radio multi-channel software-defined WMNs is design and implemented to improve failure recovery time. However, authors do not consider the computation delay and communication delay in finding rerouting paths. Authors in [15] utilize dual-band connectivity of an AP to enhance the failure recovery efficiency. However, the performance may be limited, because the AP takes care of the link failure detection and recovery in addition to packet forwarding. Further, load balance is not considered in the post-recovery network.

To provide a flexible failure recovery scheme, this paper adopts the passive strategy for the failure recovery. However, the computational complexity in calculating a rerouting path and the overhead in message exchange between the SDN controller and switches would limit the recovery performance. In [16], authors develop the local fast reroute algorithm (LFR) scheme for traffic aggregation when it occurs link failure. If link failure is detected in the LFR, all flow aggregate to a single flow and deploy rerouting paths by the SDN controller in order to minimize the utilization of flow entries. But, authors do not consider the load balance which might introduce congestion in post-recovery condition. A Priority-based flow control (PFC) is proposed in [17] by designing a variety of flow priorities to reduce the flow processing delay between the SDN controller and switches. However, LFR is not mainly designed to support reliability.

In this paper, we propose a Software Defined Fast Failure Recovery (SD-FFR) mechanism to adaptively route traffic flows in the network. With SD-FFR, the SDN controller utilizes the available link bandwidth to make routing decisions when network condition changes. Four modules, “network resource monitoring module”, “link weight management module”, “routing decision module” and “failure recovery module” are introduced and integrated carefully in SD-FFR. When a new flow joins the network, the SDN controller updates the link weights according to the available link bandwidth, before making routing decisions aiming to avoid link congestion. The SDN controller also updates the link weights after link failure recovered. Moreover, to time efficiently recover link failure, the SDN controller maintains two type of flow entries, high-priority flow entries and low-priority flow entries implying a backup routing path of each traffic flow is selected in the SDN controller beforehand. When a link fails, the SDN controller immediately provides rerouting information to the network, based on the link failure information and the high-priority flow entries. Hence, the proposed SD-FFR not only time efficiently recovers link failure but also better balances traffic loads in the network.

The remainder of this paper is organized as follows. Section II presents system architecture. Section III describes the method proposed in this paper. Section IV and V shows the implementation in commercial APs and the experimental results, respectively. We conclude this paper in Section VI.

II. SYSTEM ARCHITECTURE

A. System Description

Fig. 1 illustrates the SDN assisted wireless network in the automatic warehouse. Several SDN enabled dual-band wireless

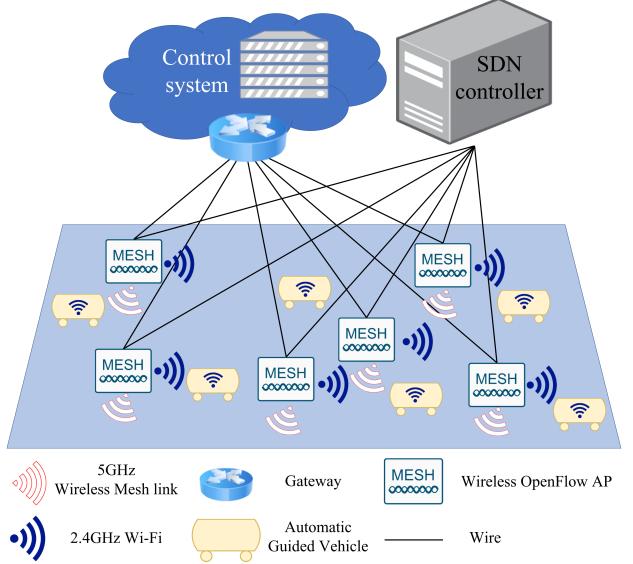


Fig. 1: Illustration of SDN assisted wireless network in the automatic warehouse.

APs are deployed in the network to provide the wireless server to the AGVs via the 2.4 GHz frequency band. All APs connect to the gateway through the Ethernet, and two APs in the network connects with each other via the 5 GHz wireless link to form a wireless mesh network. In this way, each AGV can communicate with the ICS via the gateway along the 2.4 GHz wireless link and the Ethernet. Each AGV can also communicate with others for collaboration along the 2.4 GHz wireless link and the 5 GHz wireless mesh connectivity. Moreover, following the OpenFlow protocol [18], each AP connects to the SDN controller through the Ethernet, coordinates with the SDN controller to support packet transmission in the network.

However, since AGVs may not uniformly distributed in the network, the link congestion may occur when the traffic load on a link is too heavy. In this way, the end-to-end data transmission between the AGVs and the control system may experience long delay or packet loss. On the other hand, the connection of a link may fail and then packets forwarding through the link cannot be delivered to the corresponding destinations successfully. More specifically, we focus on overcoming the following two problems in degrading the network performance in this paper:

Problem 1: Link Failure

When the Ethernet link failure occurs between the ICS and an AP, the ICS and the corresponding AGVs cannot exchange messages with each other which cause incorrect operations of the AGVs. The performance in collaboration among AGVs may degrade when the link failure occurs at a 5 GHz wireless link connecting two APs which in turn causes the loss of delivering the synchronization messages among the collaborated AGVs.

Problem 2: Link Congestion

Typically, routing paths are chosen according to the minimum number of hops. However, each AGV moves dynamically in the warehouse so that each AP may serve significantly different amount of AGVs for packet transmission. Under this circumstance, the traffic congestion may occur at some links when the load on that link is heavy. Moreover, in the presence of link failure, some flows are rerouted in order to sustain the message exchange between the ICS and some AGVs or exchange among the collaborated AGVs. As a consequence, some APs would degrade the performance in forwarding both the current traffic flows and the rerouted flows simultaneously due to the limited link bandwidth.

In this way, the goal of this paper is to utilize SDN to design a scheme to reliably and efficiently route packets in the network. To focus on addressing the above mentioned problems, we further model the network as shown in Fig. 2 and introduce several parameters to ease of this description of the rest of the paper.

B. Network Model

We consider a network model presented in Fig. 2 which is treated as an undirected graph $G(V, E)$. In the graph $G(V, E)$, we denote n as the node, any APs or the gateway (GW), V as the set of nodes, and E as the set of links. Two types of links, Ethernet and 5 GHz wireless connection are available in the topology. The GW and each AP connects with each other through the Ethernet link. Each AP connects with each other through the 5 GHz wireless link, and the 5 GHz connectivity forms a wireless mesh network. Furthermore, we denote the link to be $L(u, v) \in E$, and $u, v \in V$ denote any of the adjacent nodes in the topology. The link bandwidth at time t is denoted to be $C_{L(u,v)}^{t,b}$, where b is set to be 1 and 0, if the link is created by wireless and wired medium, respectively. Hence, $C_L^{t,1}(u, v)$ is the capacity of 5 GHz wireless link between two APs and $C_L^{t,0}(u, v)$ is the wired link between an AP and the GW. Moreover, the traffic flow sending from a source node s to a destination node d via this network is denoted as $f_{s,d}$. Since more than one traffic flows may share $L(u, v)$ to the corresponding destinations, the total amount of the traffic loads on $L(u, v)$ is denoted as $TL_{L(u,v)}^{t,b}$, where b is set to be 1 and 0 for wireless and wired link, respectively. We further denote the residual bandwidth on $L(u, v)$ to be $R_{L(u,v)}^{t,b}$, and $R_{L(u,v)}^{t,b} = C_{L(u,v)}^{t,b} - TR_{L(u,v)}^{t,b}$, where b is set to be 1 and 0, if the link is created by wireless and wired connection, respectively. Moreover, the total number of traffic flows on $L(u, v)$ is denoted as $|TL_{L(u,v)}^{t,b}|$, $b \in \{0, 1\}$. Specifically, $TL_{L(u,v)}^{t,b}$ is denoted to be the set flows routed through each link on the network.

C. Problem Formulation

To deal with the dynamical change of traffic loads as well as the link failure in the network, we design a cost function. We denote $Load(t)$ to be the maximum load of a link in the network and $RD(t)$ to be failure recovery delay. The recovery delay depends on the summation of the processing time of the

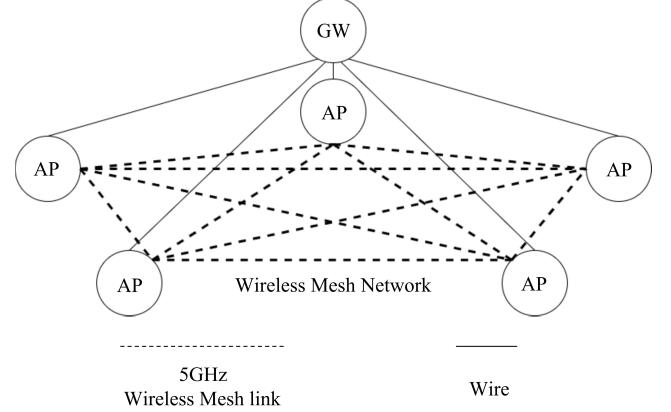


Fig. 2: Network model.

switch $H_s(t)$, the processing time of the controller $H_c(t)$, and the communication delay between the controller and switch $RTT_{s,c}(t)$. Moreover, when link failure occurs at $L(u, v)$, we set link capacity as zero, $C_{L_{sc}(u,v)}^{t,b} = 0$. Then, the total number of traffic flows and the traffic loads on $L(u, v)$ are also zero, since no flows could be forwarded on $L(u, v)$. We have $|TL_{L_{sc}(u,v)}^{t,b}| = 0$ and $TR_{L_{sc}(u,v)}^{t,b} = 0$. Consequently, the problem can be formulated as

$$\min(Load(t) + RD(t))$$

s.t.

$$Load(t) = \frac{\max |TL_L^{t,b}(u, v)|}{\text{avg}(\sum_{L(u,v) \in E} |TL_L^{t,b}(u, v)|)}, L(u, v) \in E$$

$$RD(t) = H_s(t) + H_c(t) + RTT_{s,c}(t)$$

$$C_{L(u,v)}^{t,b} \geq R_{L(u,v)}^{t,b} \geq TR_{L(u,v)}^{t,b}, \forall L(u, v) \in E, b \in \{0, 1\}$$

The objective function $\min(Load(t) + RD(t))$ aims at minimizing the maximal traffic loads on a link and the failure recovery delay. Thus, traffic flows would be time efficiently rerouted without introducing additional link congestion, when a traffic flow joins the network or a link failure occurs at time t .

Additionally, TABLE I summarizes the definition of all the network parameters using in this paper.

III. SOFTWARE DEFINED FAST FAILURE RECOVERY(SD-FFR) MECHANISM

In this paper, we utilize SDN and then propose the Software Defined Fast Failure Recovery (SD-FFR) mechanism to time efficiently react to the change of network status by conditionally reroute traffic flows. With SD-FFR, the performance degradation resulted from link failure and link congestion in the SDN assisted wireless network can be solved. In this way, the reliability and load balance can be properly sustained in the network.

Fig. 3 shows the architecture of integrating the proposed SD-FFR mechanism into existing SDN architecture. Specifically, the proposed SD-FFR mechanism is implemented in

TABLE I: Parameter

Notation	Description
$G(V, E)$	The graph of the network topology.
$L(u, v)$	The link between node u and node v .
$TR_{L(u,v)}^{t,b}$	Traffic loads pass through $L(u, v)$ at t time instant.
n	denote the OpenFlow device.
$L_{sc}(u, v)$	The link with state change between node u and v .
b	b is a Binary value, $b = 1$ denote the wireless mesh link and if $b = 0$ denote the Ethernet link.
$C_{L(u,v)}^{t,b}$	C_1 denote the Wireless Mesh link capacity and C_0 denote the Ether link capacity at t time instant.
$R_{L(u,v)}^{t,b}$	Residual link capacity of $L(u, v)$ at t time instant.
$W_{L(u,v)}$	The weight of $L(u, v)$.
$W_{L_{sc}(u,v)}$	The weight of state change $L(u, v)$.
Q_b	Q_1 denote the adjustable value for Wireless Mesh link, and Q_0 denote the adjustable value for Ether link.
$f_{s,d}$	Traffic demand from node s to node d .
$P_{f_{s,d}}$	Path of traffic demand $f_{s,d}$.
$TL_{L(u,v)}^{t,b}$	The set of $f_{s,d}$ pass through the link $L(u, v)$ at t time instant.
$AL_{L_{sc}(u,v)}^{t,b}$	The set of $f_{s,d}$ affected by fault pass through the link $L_{sc}(u, v)$ at t time instant.
$FR_{P_H}^{f_{s,d}}$	High-priority forwarding rule.
$FR_{P_L}^{f_{s,d}}$	Low-priority forwarding rule.

the SDN controller. Following the OpenFlow protocol, the network status can be delivered through the Southbound API from the APs to the controller. The routing decisions are also delivered from the controller to the APs via the Southbound API. In this way, packets can be adaptively routed in the network according to the network conditions. As illustrated in Fig. 3, four modules consist of the SD-FFR mechanism, and they are “network resource monitoring module”, “link weight management module”, “routing decision module” and “failure recovery module”.

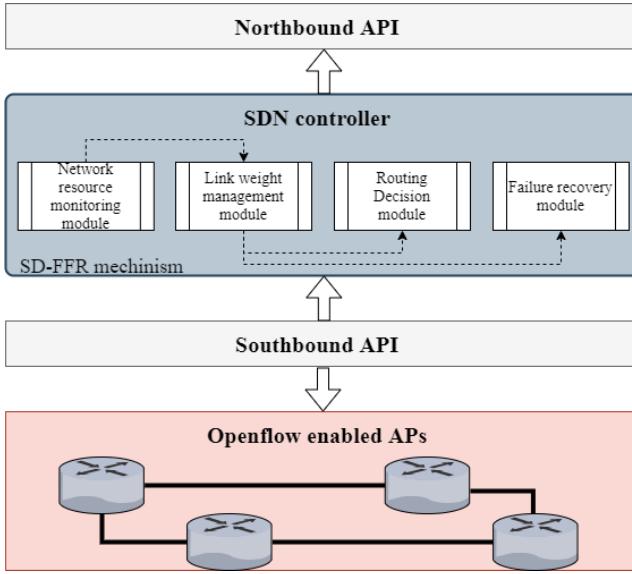


Fig. 3: The architecture of SD-FFR mechanism.

A. Network Resource Monitoring Module

To effectively utilize the available link bandwidth in the network, the “network resource monitoring module” collects the information of the available link bandwidth in data plane, when a traffic flow $f_{s,d}$ joins the network. Then, the collected information is passed to the “link weight management module” for further processing.

Alg. 1 describes the operation of the “network resource monitoring module”. The main purpose of Alg. 1 is to update the current residual link bandwidth, denoting as $R_{L(u,v)}^{t,b}$, by collecting the traffic loads on each link, denoting as $TR_{L(u,v)}^{t,b}$, of the network, when a new flow joins the network. In this way, the updated $R_{L(u,v)}^{t,b}$ can be used in “routing decision module” for routing decision making. In Alg. 1, when a new flow joins the network.

Alg. 1: Network Resource Monitoring Module

- 1: SDN controller monitors $TR_{L(u,v)}^{t,b}$, $\forall L(u, v) \in E$.
 - 2: **If** $f_{s,d}$ joins the network **then**
 - 3: **Forall** $\forall L(u, v) \in E$ **do**
 - 4: $R_{L(u,v)}^{t,b} := C_{L(u,v)}^{t,b} - TR_{L(u,v)}^{t,b}$
 - 5: **end**
 - 6: **Back to line 1**
-

B. Link Weight Management Module

To balance traffic loads in the network, the link weight management module maintains the weight of each link. Specifically, we introduce two parameters, $W_{L(u,v)}$ and $W'_{L(u,v)}$, which are denoted to be the weights in the normal network state and the post-recovery network state, respectively. The normal network state implies no link failure occurs while the post-recovery network state implies the flows forwarded through failed links have been recovered. Furthermore, $W_{L(u,v)}$ is updated according to the link capacity usage, and $W'_{L(u,v)}$ is adjusted based on traffic demands on each link.

- Link weight in normal network state:

$$W_{L(u,v)} := Q_b \times e^{\frac{\alpha(C_{L(u,v)}^{t,b} - R_{L(u,v)}^{t,b})}{C_{L(u,v)}^{t,b}}}, \quad b \in \{0, 1\} \quad (1)$$

The link weight in the normal network state is calculated according to (1). The link weight is applied for making routing decisions to balance traffic flows and prevent traffic congestion in the network. Hence, the available bandwidth on each link, $\frac{C_{L(u,v)}^{t,b} - R_{L(u,v)}^{t,b}}{C_{L(u,v)}^{t,b}}$. Further, two parameters Q_b and α are set to increase the probability in selecting the wired link for data transmission aiming to utilize the large link bandwidth of the Ethernet which is usually larger than the 5 GHz wireless link. The Q_b is in further set as $0 < 2Q_b < 1$ and $Q_0 + Q_1 = 1$.

- Link weight in post-recovery network state:

$$W'_{L(u,v)} := e^{|TL_{L(u,v)}^{t,b}|} \quad (2)$$

The link weight in the post-recovery is calculated according to (2). The link weight is applied for making routing decisions to further balance traffic loads, after the link failure recovery. To time efficiently adjust the link weight to assist the fast failover and the load balance, only total number of traffic demands on each link, $|TL_{L(u,v)}^{t,b}|$, is considered for the link weight calculation, instead of considering the traffic loads on each link as (1). In this way, not only the time spending in calculating the link weight can be reduced, the time spending in exchanging control messages between APs and the controller can be avoided.

The detail operation of the “link weight management module” is described in Alg. 2. When the network starts, the link weight, $W_{L(u,v)}$, is initialized as Q_b . As long as a new flow joins the network, $W_{L(u,v)}$ is updated according to the link bandwidth utilization. On the other hand, the link weight after link failure recovery, $W'_{L(u,v)}$, is calculated based on the number of flows on $L(u,v)$.

Alg. 2: Link Weight Management Module

```

1: If Network initialization then
2:    $W_{L(u,v)} := Q_b$ 
3: Else if Normal network state then
4:    $W_{L(u,v)} := Q_b \times \exp\left(\frac{\alpha(C_{L(u,v)}^{t,b} - R_{L(u,v)}^{t,b})}{C_{L(u,v)}^{t,b}}\right)$ 
5: Else if Post-recovery network state then
6:    $W'_{L(u,v)} := \exp(|TL_{L(u,v)}^{t,b}|)$ 
end
```

C. Routing Decision Module

When the traffic demands join the network, the link weights are firstly updated, and then the “routing decision module” adjusts the routing path based on the Dijkstra algorithm [19] as well as installs the forwarding rule to the corresponding APs to create the routing path for the new joining flow $f_{s,d}$.

Alg. 3 describes the process of “routing decision module”. When $f_{s,d}$ joins the network, considering the updated link weights through the Alg. 2, the routing path $P_{f_{s,d}}$ in forwarding data of $f_{s,d}$ is calculated through the Dijkstra algorithm. Then, the SDN controller adds the low-priority forwarding rule $FR_{P_L}^{f_{s,d}}$ to the corresponding nodes on path $P_{f_{s,d}}$. Moreover, this algorithm maintains the set flows routed through each link on the network which is denoted as $TL_{L(u,v)}^{t,b}$. Specifically, $TL_{L(u,v)}^{t,b}$ is managed to assist the operation of “failure recovery module”.

D. Failure Recovery Module

The “failure recovery module” takes care of adjusting the link weights based on (2) to find backup path and executing fast rerouting to time efficiently recover the data transmission of the affected flows, when link failure occurs. Additionally, the “failure recovery module” carefully sets the priority of flow entries and the timing installing or removing forwarding rules to improve the recovery efficiency. Specifically, two sets are maintained in this module.

Alg. 3: Routing Decision Module

```

1: Input: Traffic demand  $f_{s,d}$ 
2: Output: Routing decision
3: If  $f_{s,d}$  joins the network then
4:   Forall  $L(u,v) \in E$  do
5:     Run Alg. 2
6:   end
7:   Find the shortest path  $P_{f_{s,d}}$  on  $G(V, E)$ .
8:   Install rule  $FR_{P_L}^{f_{s,d}}$  to  $n$ ,  $\forall n \in P_{f_{s,d}}$ 
9:    $TL_{L(u,v)}^{t,b} := TL_{L(u,v)}^{t,b} \cup f_{s,d}, \forall L(u,v) \in P_{f_{s,d}}$ 
10:  Return  $P_{f_{s,d}}$ 
11: Else if  $f_{s,d}$  leaves the network then
12:    $TL_{L(u,v)}^{t,b} := TL_{L(u,v)}^{t,b} \setminus f_{s,d}, \forall L(u,v) \in E$ 
end
```

1. $TL_{L(u,v)}^{t,b}$: We denote $TL_{L(u,v)}^{t,b}$ as the set of traffic flows on the $L(u,v)$ in the network. The set is updated as long as a traffic flow joins or leaves the network. Also, it is updated when the path of a traffic flow changes.
2. $AL_{L_{sc}(u,v)}^{t,b}$: We denote $AL_{L_{sc}(u,v)}^{t,b}$ as the set of traffic flows on the failure link $L_{sc}(u,v)$ in the network. $AL_{L_{sc}(u,v)}^{t,b}$ records the traffic flows that need to be rerouted due to the occurrence of the link failure. Specifically, when the controller receives a packet with a **LINK REMOVED** message, it records the traffic demand affected by the failure link $L_{sc}(u,v)$. The set is also updated when the controller receives a packet with a **LINK ADD** message.

Alg. 4 describes the operation of the “failure recovery module”. When the controller receives a **LINK REMOVED** message, it gets the information of the traffic flows affected by $L_{sc}(u,v)$ in the $TL_{L(u,v)}^{t,b}$. Then, the weight of $L_{sc}(u,v)$ is set to be infinity. Before calculating a backup path for each traffic flow affected by a failed link, the weights W' for links other than the failed link based on (2) so that a lightly loaded backup path can be picked for rerouting. Further, the SDN controller installs the high priority flow entries $FR_{P_H}^{f_{s,d}}$ on the nodes of the backup path aiming to replace the low priority flow entries $FR_{P_L}^{f_{s,d}}$. Then, two corresponding sets $TL_{L(u,v)}^{t,b}$ and $AL_{L_{sc}(u,v)}^{t,b}$ are updated in further. Moreover, the SDN controller directly installs the high priority flow entries, denoted as $FR_{P_H}^{f_{s,d}}$. In this way, the latency of transmitting the **Delete** flow operation message can be reduced. The delay in installing new flow entries can also be reduced because the SDN controller does not need to transmit the **Modify** flow operation message and the communication in obtaining flow entries information from the nodes in the network can be avoided.

When the controller receives the packet with **LINK ADD** message, it checks $AL_{L_{sc}(u,v)}^{t,b}$ to find the traffic flows routed through their backup paths and then update the residual bandwidth of those flows by Alg. 1. As a result, those flows can be rerouted by executing Alg. 3. Furthermore, both the $TL_{L(u,v)}^{t,b}$ and $AL_{L_{sc}(u,v)}^{t,b}$ are updated, and the $FR_{P_H}^{f_{s,d}}$ of the corresponding nodes are detected. Note that the priority of the new flow entries $FR_{P_L}^{f_{s,d}}$ is lower than the priority of backup

flow entries $FR_{P_H}^{f_{s,d}}$. Therefore, $f_{s,d}$ is firstly matched by the backup flow entries, $FR_{P_H}^{f_{s,d}}$, before the $FR_{P_H}^{f_{s,d}}$ is deleted. After the installation of the low priority flow entries, $FR_{P_L}^{f_{s,d}}$, on the corresponding node in the network, the unnecessary backup flow entries, $FR_{P_H}^{f_{s,d}}$, are deleted. Through this process, the forwarding path can be switched without the delay of transmitting delete flow instruction from the SDN controller to the corresponding nodes in the network. More specifically, the process of **LINK ADD** makes routing decisions with the assistance of Alg.1 and Alg.3 without worrying about the reroute speed and the communication delay between the controller and the switch. In this way, the delay in message exchange between the controller and the APs as well as in computing alternative routes can be decreased. Further, the efficiency of load balancing can also be improved.

Alg. 4: Failure Recovery Module

```

1: Receive a OFPT PORT STATUS packet.
2: If the link state is LINK REMOVED then
3:   Use  $L_{sc}(u, v)$  to find all the affected traffic demand
 $f_{s,d}$  in  $TL_{L(u,v)}^{t,b}$ 
4:    $W_{L_{sc}(u,v)} := \infty$ 
5:    $W'_{L_{sc}(u,v)} := \infty$ 
6:   Forall the affected traffic demand  $f_{s,d}$  pass through
 $L_{sc}(u, v)$  do
7:     Forall  $L(u, v) \in E \setminus L_{sc}(u, v)$  do
8:       Run Alg. 2
end
9:   Use Dijkstra algorithm to find the backup path
 $P_{f_{s,d}}$  on  $G(V, E)$  base on  $W'$ 
10:  Install  $FR_{P_H}^{f_{s,d}}$  to  $n$ ,  $\forall n \in P_{f_{s,d}}$ 
11:   $AL_{L_{sc}(u,v)}^{t,b} := AL_{L_{sc}(u,v)}^{t,b} \cup f_{s,d}$ 
12:   $TL_{L(u,v)}^{t,b} := TL_{L(u,v)}^{t,b} \setminus AL_{L_{sc}(u,v)}^{t,b} \cup f_{s,d}$ ,  $\forall$ 
 $L(u, v) \in P_{f_{s,d}}$ 
end
13: Else
14:   Use  $L_{sc}(u, v)$  to find all the affected traffic demand
 $f_{s,d}$  in  $AL_{L_{sc}(u,v)}^{t,b}$ 
15:   Forall the affected traffic demand  $f_{s,d}$  pass through
 $L_{sc}(u, v)$  do
16:     Run Alg. 1 (Line 3-8)
17:     Run Alg. 3 (Line 1-9)
18:      $AL_{L(u,v)}^{t,b} := AL_{L(u,v)}^{t,b} \setminus f_{s,d}$ 
19:      $TL_{L(u,v)}^{t,b} := TL_{L(u,v)}^{t,b} \setminus f_{s,d}$ ,  $\forall L(u, v) \in E$ 
20:      $TL_{L(u,v)}^{t,b} := TL_{L(u,v)}^{t,b} \cup f_{s,d}$ ,  $\forall L(u, v) \in P_{f_{s,d}}$ 
21:     Forall  $n$ ,  $\forall n \in V$  do
22:       Delete  $FR_{P_H}^{f_{s,d}}$  in node  $n$ 
end
end
end
```

E. Description of the SD-FFR Mechanism

After properly integrating the four algorithms of the proposed SD-FFR mechanism, Fig. 4 depicts the flowchart of the SD-FFR. Specifically, when the network starts, the SD-FFR

mechanism initializes the link weights by Alg. 2. In addition, the Alg. 2 is executed to manage the link weights in the network. Following the Alg. 2, information of residual link bandwidth is updated by Alg. 1. Simultaneously, the routing decisions can be made through Alg. 3, when a traffic flow joins the network. When the failure occurs, SD-FFR performs Alg. 4 for failure recovery processing.

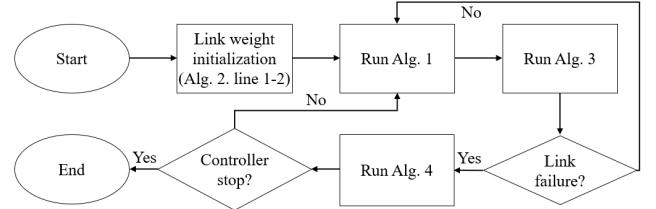


Fig. 4: The flowchart of SD-FFR mechanism.

IV. IMPLEMENTATION

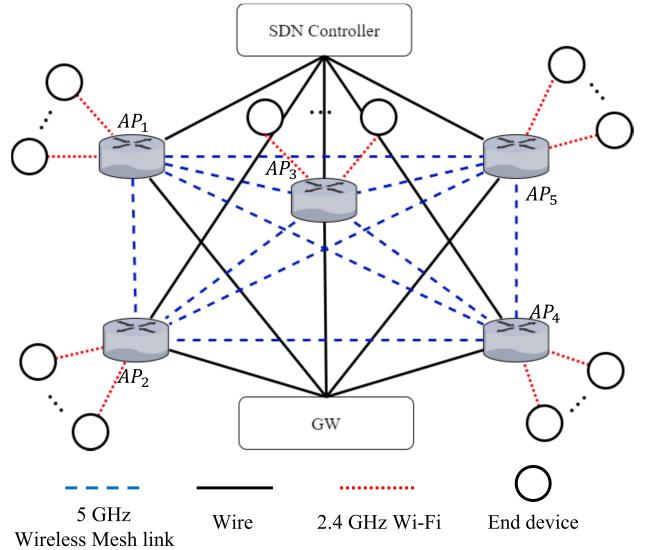


Fig. 5: Network environment.

In this section, we evaluate the behavior of data transmission in SDN assisted dual-band WiFi networks by commercial APs, and apply the measurement for Mininet simulation setup in Section V aiming to effectively validate the performance of the proposed SD-FFR in real network environment. As a result, following Fig. 5, each commercial AP has Ethernet connectivity to the Internet and the SDN controller, 5 GHz wireless mesh connectivity to neighboring APs and a 2.4 GHz connectivity to mobile devices. Further, we enable OpenFlow1.3 [18] protocol on the commercial APs (i.e., TP-Link Archer C5 AC175) by installing both OpenWrt and OpenvSwitch (OVS) on them. Then, the OVS running on the APs can communicate with the ONOS controller. We use iPerf and Ping to measure the link bandwidth and data transmission latency under different network conditions. Specifically, we collect measurements by using wired and wireless link individually which can be further applied for the experimental setup in Mininet.

TABLE II presents the measured link bandwidth and data transmission delay in the network formed by the commercial APs. As shown in TABLE II, Ethernet provides the largest link bandwidth (i.e., 100 Mbps) and the smallest data transmission delay (i.e., 0.37ms). 71 Mbps and 47 Mbps are measured through the 5 GHz wireless mesh link and the 2.4 GHz link, respectively. Unsurprisingly, the delay over 5 GHz wireless link is smaller than the 2.4 GHz wireless link. The results in TABLE II will be applied in next section to investigate the network performance in the presence of link failure.

TABLE II: Mininet modeling.

	Link bandwidth	Delay
Wired link	100 Mbps	0.37 ms
5 GHz wireless mesh link	71 Mbps	0.66 ms
2.4 GHz wireless link	47 Mbps	1.8 ms

A. The Challenge in Link Failure Detection by OVS on Commercial APs

To investigate the behavior of link failure detection and recovery on the commercial APs, we manually unplug the Ethernet on an AP which in turn cannot access to the Internet. We found that the OVS cannot instantly receive the link failure information because the synchronization between the OVS and the operating system of the AP is non-realtime. On average, additionally 8 seconds delay is introduced for the ONOS to receive the failure message. However, this situation does not happen in the Mininet.

Furthermore, we design the “network state detection acceleration module” shown in Fig. 6 executing on the data plane, to mitigate the adverse impact on network performance. Specifically, the AP periodically detects its ports status. When the AP detects the port status changes, the OVS receives Delete or Add command. Fig. 7 shows 25 measurements of the delay for the AP in detecting whether the port state changes. As shown in Fig. 7, the average delay reduces to 44.6 ms by implementing the “network state detection acceleration module” on the AP. Then, considering the minimized response time of link failure, we input the 44.6ms delay into the Mininet to evaluate the failure recovery performance of the proposed SD-FFR in the real network environment. Additionally, with the assistance of the “network state detection acceleration module”, the proposed SD-FFR can support failure recovery more time efficiently in real network environment.

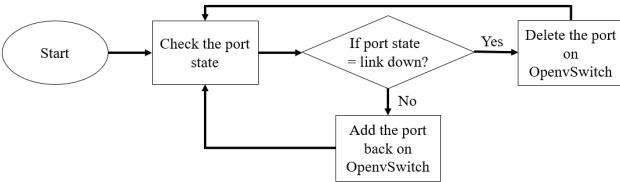


Fig. 6: The flowchart of network state detection acceleration module.

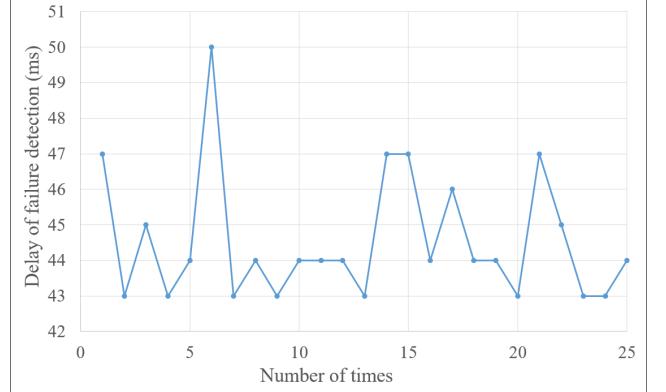


Fig. 7: Delay of failure detection.

V. EVALUATION RESULTS

A. Experimental Setup

To evaluate the performance of the proposed SD-FFR, we use Mininet to create a network topology present in Fig. 5. Specifically, the Mininet runs on the computer with Intel i7-8705G CPU and 16 GB RAM. In the network, five APs not only directly connect to the GW through wired connection but also connect with neighboring APs via 5 GHz wireless link. Each AP serves provides end devices wireless connectivity operated at the 2.4 GHz frequency band. Specifically, we adopt the parameters measured in the commercial WiFi APs shown in TABLE II and the 44.6 ms link failure detection latency described in Section IV-A to configure the network.

Several experiments are conducted to validate the effectiveness of the SD-FFR. The goal of the evaluation are twofolds. First of all, we evaluate the impact of the parameters Q_b and α of (1) on the network performance in terms of load balance. Furthermore, we investigate the impact network conditions on packet loss rate and throughput. The method in [15] is applied for the comparison, because it considers similar network setup.

B. The Impact of Parameters on Load Balance

In this experiments, different combinations of Q_b and α are set to investigate the relationships between the link weight function (1) and the achieved load balance in the normal network state. Sixteen hosts are introduced on AP_1 and AP_2 in the network. Each host on AP_1 performs an iPerf client to communicate with the corresponding iPerf server connecting with AP_2 , and then sixteen UDP traffic flows constantly transmit packets with 20 Mbps data rate to the network. With appropriately picked Q_b and α , each WMN link should share three traffic flows while the Ethernet connection handles four traffic flows to avoid link congestion, because each AP equips with one Ethernet and four WMN links, TABLE III illustrates the number of traffic flows which can be successfully delivered under different pairs of Q_b and α for the performance evaluation. As present in TABLE III, when $\alpha = 10$ and $Q_b \in \{0.1, 0.3, 0.5, 0.7\}$ SD-FFR can handle 16 traffic demands. Therefore, in the rest of the experiments, we set $\alpha = 10$ and $Q_0 = 0.1$ in calculating the link weights under the normal network state.

TABLE III: The adjustable parameters Q and α correspond to the number of successfully process traffic demand.

$Q_0 \backslash \alpha$	10	20	30	40	50
0.1	16	13	10	9	9
0.3	16	13	10	9	7
0.5	16	13	12	10	7
0.7	16	13	12	9	7
0.9	15	13	12	10	7

We set Q_0 to 0.1 and α to 10 to load balance efficiency in the post-recovery network, according to TABLE III, $0 < 2Q_0 < 1$ and $Q_0 + Q_1 = 1$. Moreover, instead of introducing 16 iPerf server-client pairs in the previous experiment, only twelves UDP traffic demands whose individual data rate is 20 Mbps. We disconnect the direct Ethernet connectivity between AP_1 and GW as link failure in order to study the performance of the post-recovery network. TABLE IV illustrates the routing path of each traffic demand before and after fault recovery. As found in TABLE IV, the traffic demand 1, 3 and 7 are rerouted through AP_3 , AP_4 and AP_5 , respectively because the link between AP_1 and GW fails. These three flows are evenly redirected through three different paths to avoid the link congestion after failure recovery.

TABLE IV: Load-balance in post-recovery network.

Traffic demand	Original path	Path of post recovery network
1	$AP_1 - GW - AP_2$	$AP_1 - AP_3 - AP_2$
2	$AP_1 - AP_2$	$AP_1 - AP_2$
3	$AP_1 - GW - AP_2$	$AP_1 - AP_4 - AP_2$
4	$AP_1 - AP_3 - AP_2$	$AP_1 - AP_3 - AP_2$
5	$AP_1 - AP_4 - AP_2$	$AP_1 - AP_4 - AP_2$
6	$AP_1 - AP_5 - AP_2$	$AP_1 - AP_5 - AP_2$
7	$AP_1 - GW - AP_2$	$AP_1 - AP_5 - AP_2$
8	$AP_1 - AP_2$	$AP_1 - AP_2$
9	$AP_1 - AP_2$	$AP_1 - AP_2$
10	$AP_1 - AP_3 - AP_2$	$AP_1 - AP_3 - AP_2$
11	$AP_1 - AP_4 - AP_2$	$AP_1 - AP_4 - AP_2$
12	$AP_1 - AP_5 - AP_2$	$AP_1 - AP_5 - AP_2$

C. The Impact of Network Conditions on Performance

TABLE V: Mean of packet loss rate (%).

	10Mbps	15Mbps	20Mbps	25Mbps
Case1	0.39	0.39	0.81	1.28
Case2	0.39	0.42	0.60	1.35

1) *The impact of flow rates on packet loss:* In this experiment, we consider a traffic flow sent through AP_1 and AP_2 to the destination. We used iPerf to generate UDP traffic of 10Mbps, 15Mbps, 20Mbps, and 25Mbps for 30 seconds, respectively and we repeat the measurement 100 times per flow rate aiming to measure the packet loss rate in the presence of link failure. Further, two cases of failure, wired and wireless link failure, are considered. We denote the link failure occurs in the wired connection as case 1 while case 2 denotes the wireless link failure. Link failure occurs at the 15th second and only occurs once in each trial. Fig. 8 - 9 present the packet loss rate during failure recovery in case 1 and 2 respectively.

The x-axis shows the number of repeated trials and the y-axis indicates the corresponding average packet loss rate. The packet loss rarely occurs when flow rates are smaller than 20 Mbps in both cases. With the increasing flow rates, the packet loss rate increases, and the traffic flow with 25 Mbps data rate approximately experience at most 5% packet loss. However, on average, the packet loss rate for 25 Mbps data rate is about 1.3% only as shown in TABLE V. Since 5 GHz wireless link provides 71 Mbps link bandwidth, the average packet loss rate in both cases are similar as concluded in TABLE V. Specifically, SD-FFR time efficiently recovers link failure by rerouting affected traffic flows and carefully plans the operation of failover to reduces the recovery delay. Therefore, the packet loss rate is low.

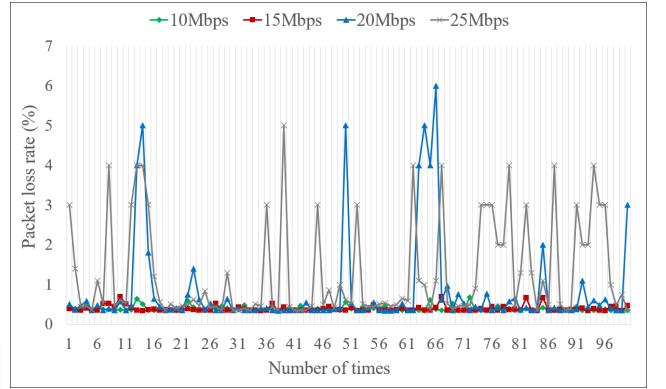


Fig. 8: packet loss rate in case 1.

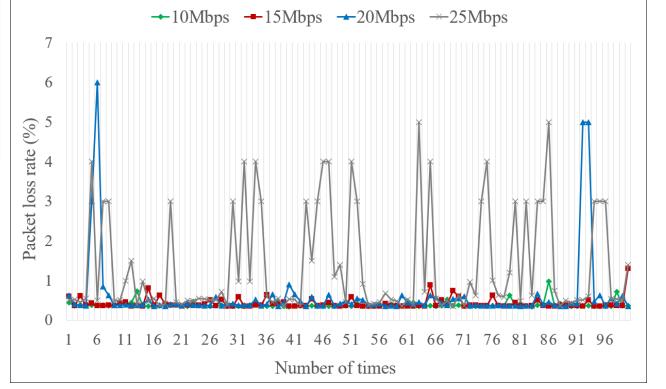


Fig. 9: Packet loss rate in case 2.

2) *The Impact of Link Failure Recovery on TCP Throughput :* In this experiment, we evaluate the impact of link failure recovery on TCP traffic flow following the previous setup. Hence, the link also fails at the 15th second in both cases. The obtained TCP throughput and congestion windows in case 1 are shown in Fig. 10 - 13. Similarly, Fig. 10 - 13 depicts the TCP throughput and the congestion windows in case 2.

It can be found that both the congestion window size and the TCP throughput drops significantly at the 15th second when a link fails in the network. However, the proposed SD-FFR time efficiently detects the link failure and reroutes the traffic by install the high priority of flow entries during

failover. Consequently, the end-to-end performance can be fast recovered.

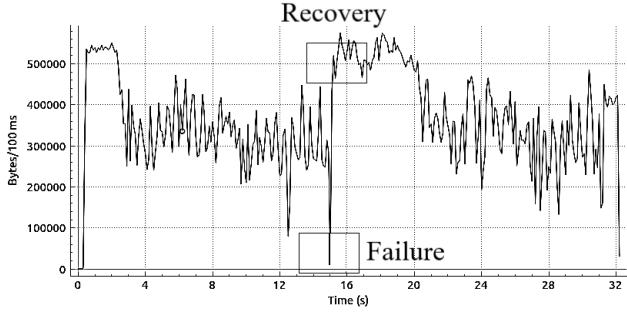


Fig. 10: TCP Throughput in case 1.

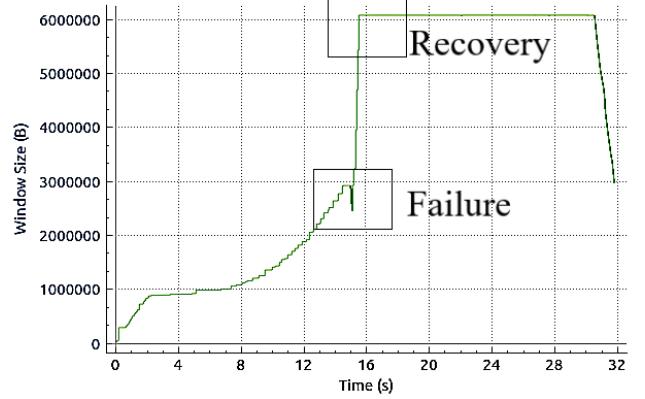


Fig. 13: Congestion window size in case 2.

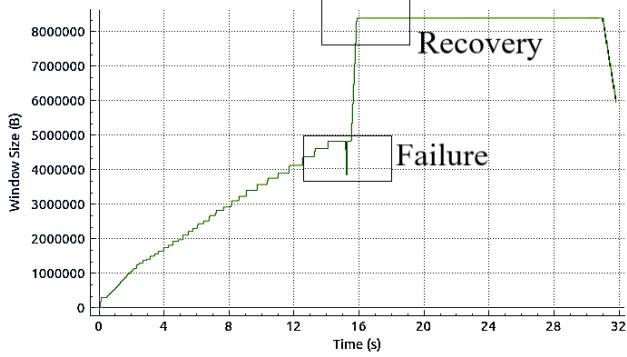


Fig. 11: Congestion window size in case 1.

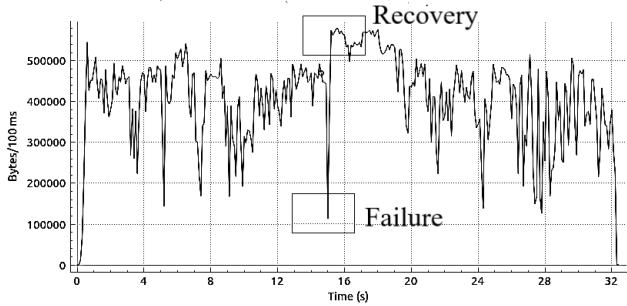


Fig. 12: TCP throughput in case 2.

D. The impact of failure recovery schemes on network performance

In this experiment, we evaluate the performance of failure recovery through our proposed SD-FFR and SCONN [15] by using Ping. We introduce the link failure at the 15th second within the 30 seconds experiment, and then repeat the experiment 100 times. Fig. 14 and Fig. 15 plot the link failure recovery time for each trial in case 1 and 2, respectively. The mean and variance of the recovery time are summarized in TABLE VI. The proposed SD-FFR recovers the link failure 18 times faster comparing to SCONN in both cases, since SD-FFR takes less than 60 ms to complete failure recovery

while SCONN takes at 1000 ms for failure recovery. It is because SD-FFR carefully plans the priority of flow entries during failover and performs the installation and deletion of forwarding rules at the right time. The time in collecting link usage and calculating rerouting paths can be greatly saved which in turn reduces the recovery delay significantly. SCONN does not have these considerations, and thus the recovery time is high.

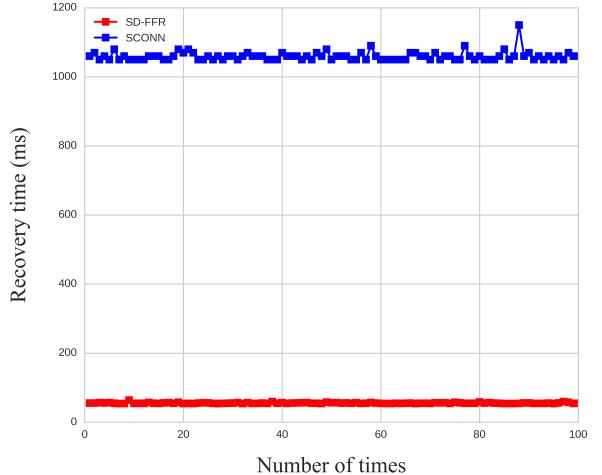


Fig. 14: Failure recovery time in case 1.

TABLE VI: Compare between different approaches on failure recovery time.

	Case 1	Case 2
SD-FFR	Mean: 55.5ms STD: 1.48	Mean: 57.6ms STD: 1.59
SCONN [15]	Mean: 1060ms STD: 0.01	Mean: 1061ms STD: 0.01

VI. CONCLUSION

In this paper, we propose a SD-FFR mechanism to improve the network reliability in the automatic warehouse. With SD-FFR, the SDN controller reroutes the traffic to the backup

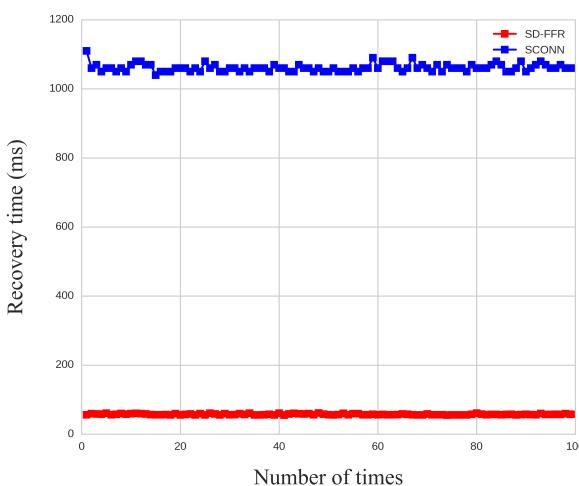


Fig. 15: Failure recovery time in case 2.

wireless mesh network to react to the occurrence of link failure. Specifically, SD-FFR carefully sets the priority of the flow entries on each nodes in the network to minimize the appropriate operation time installing and deleting the corresponding forwarding rules of the traffic flows influenced by the link failure. Therefore, the link failure can be time efficiently recovered. Considering that the traffic load in the automatic warehouse may be unevenly balanced, the SD-FFR mechanism also design strategy in balancing traffic loads no matter link failure happens or not.

We evaluate the proposed SD-FFR mechanism on Mininet, and the results show the effectiveness of SD-FFR in improving the network performance. Specifically, in the Mininet evaluations, the parameters of link bandwidth and latency in recovering link failure are configured, according to the measurement by commercial APs to study the impact of applying SD-FFR in the real network. In the future, we will design a hybrid failure recovery scheme integrating the active and passive failure recovery strategies in order to improve our proposed mechanism further.

REFERENCES

- [1] Y. Liu, K. F. Tong, X. Qiu, Y. Liu, and X. Ding, “Wireless Mesh Networks in IoT networks,” in *2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition*, May. 2017, pp. 183–185.
- [2] A. Karaagac, J. Haxhibeqiri, W. Joseph, I. Moerman, and J. Hoebeke, “Wireless industrial communication for connected shuttle systems in warehouses,” in *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, May. 2017, pp. 1–4.
- [3] E. Alizadeh Jarchilo, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, “To Mesh or not to Mesh: Flexible Wireless Indoor Communication Among Mobile Robots in Industrial Environments,” vol. 9724, Jul. 2016, pp. 325–338.
- [4] A. Fellan, C. Schellenberger, M. Zimmermann, and H. D. Schotten, “Enabling Communication Technologies for Automated Unmanned Vehicles in Industry 4.0,” in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2018, pp. 171–176.
- [5] Q. Chen, X. J. Zhang, W. L. Lim, Y. S. Kwok, and S. Sun, “High Reliability, Low Latency and Cost Effective Network Planning for Industrial Wireless Mesh Networks,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2354–2362, Dec. 2019.
- [6] K. C. Karthika, “Wireless mesh network: A survey,” in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Mar. 2016, pp. 1966–1970.
- [7] K. Bao, J. D. Matyjas, F. Hu, and S. Kumar, “Intelligent Software-Defined Mesh Networks With Link-Failure Adaptive Traffic Balancing,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 266–276, Jun. 2018.
- [8] W. J. Lee, J. W. Shin, H. Y. Lee, and M. Y. Chung, “Testbed implementation for routing WLAN traffic in software defined wireless mesh network,” in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, Jul. 2016, pp. 1052–1055.
- [9] P. Patil, A. Hakiri, Y. Barve, and A. Gokhale, “Enabling Software-Defined Networking for Wireless Mesh Networks in smart environments,” in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, Oct. 2016, pp. 153–157.
- [10] M. Labraoui, M. M. Boc, and A. Fladenmuller, “Software Defined Networking-assisted routing in wireless mesh networks,” in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Sep. 2016, pp. 377–382.
- [11] A. Detti, C. Pisa, S. Salsano, and N. B. Melazzi, “Wireless Mesh Software Defined Networks (wmSDN),” in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2013, pp. 89–95.
- [12] S. Feng, Y. Wang, X. Zhong, J. Zong, X. Qiu, and S. Guo, “A ring-based single-link failure recovery approach in SDN data plane,” in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Jul. 2018, pp. 1–7.
- [13] G. Saldamli, H. Mishra, N. Ravi, R. R. Kodati, S. A. Kuntamukkala, and L. Tawalbeh, “Improving link failure recovery and congestion control in SDNs,” in *2019 10th International Conference on Information and Communication Systems (ICICS)*, Jun. 2019, pp. 30–35.
- [14] M. Tanha, D. Sajjadi, and J. Pan, “Demystifying Failure Recovery for Software-Defined Wireless Mesh Networks,” in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 488–493.
- [15] C. Huang, W. Chung, and C. Liu, “SCONN: Design and Implement Dual-Band Wireless Networking Assisted Fault Tolerant Data Transmission in Intelligent Buildings,” in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Aug. 2018, pp. 1–5.
- [16] X. Zhang *et al.*, “Local Fast Reroute With Flow Aggregation in Software Defined Networks,” *IEEE Communications Letters*, vol. 21, no. 4, pp. 785–788, Dec. 2017.
- [17] B. Oh, S. Vural, N. Wang, and R. Tafazolli, “Priority-Based Flow Control for Dynamic and Reliable Flow Management in SDN,” *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1720–1732, Nov. 2018.
- [18] M. Nick *et al.*, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
- [19] D. B. Johnson, “A note on dijkstras shortest path algorithm,” *J. ACM*, vol. 20, no. 3, p. 385388, Jul. 1973.