Grenze International Journal of Engineering and Technology, January Issue

Optimized SDN Resilience: A Novel Flow Aggregation Approach for Ultra-Fast Network Recovery

Nazir Mohammed
ASP WEB SOLUTIONS LLC, India
Email: nazir.ms@gmail.com

***Abstract*— Software-Defined Networking (SDN) introduces a paradigm where the control and data planes are separated, network intelligence is centralized, and the physical network infrastructure is abstracted from applications. While SDN en- hances network management capabilities in carrier-grade net- works (CGNs), ensuring resiliency remains a challenge, especially in the presence of device and link failures. Network component failures can disrupt traffic forwarding, leading to packet loss, service unavailability, and degraded Quality of Service (QoS). Therefore, efficient failure detection and recovery mechanisms are critical for maintaining seamless service.**

I. INTRODUCTION

Software Defined Networking (SDN) simplifies fault man- agement in contrast to conventional distributed architectures by enabling switches to delegate recovery responsibilities to a central controller. Upon failure detection, the controller re- calculates alternate forwarding paths and disseminates updated flow rules. However, such centralization may introduce latency and burden the control plane, especially during extensive recovery processes.

To overcome these challenges, pre-establishing backup routes within switches can reduce dependency on the con- troller, accelerate failover, and conserve scarce TCAM space. A synthesis of prior research reveals several critical objectives for robust recovery strategies in SDN environments:

- Establishment of backup paths prior to failures
- Efficient memory allocation for path entries
- In-switch rerouting capabilities for broken links
- Reduced communication overhead with the control layer

This study introduces two failure handling techniques—Local Immediate (LIm) and Immediate Controller Dependent (ICoD)—that facilitate fast and lightweight recovery by employing aggregation-based strategies. The proposed mechanisms implement group-based forwarding, where multiple flows sharing the same egress port are redirected through unified group entries. These entries autonomously detect link disruptions and initiate detours via predesignated alternatives.

Additionally, flow entries are marked with upstream link metadata, enabling path redirection without assigning unique backup routes per flow. LIm harnesses OpenFlow's native fast failover group functionality, enabling purely local rerouting without controller intervention. Conversely, ICoD leverages

Indirect groups to maintain compatibility across a broader set of switches, requiring only selective controller updates when needed.

Grenze ID: 01.GIJET.12.1.159

These approaches are tailored to meet the sub-50 ms recovery benchmark mandated in carrier-grade network (CGN) environments, ensuring swift fault recovery while minimizing stress on network control and switching infrastructure. A performance comparison is conducted against a standard controller-based (CoD) recovery technique to validate effectiveness.

II. Related Work

Resilience in network systems has received widespread attention across multiple research domains. In the context of profiling and system-level optimization, Jyoti Singh [1] introduced IHPP, a dynamic binary instrumentation tool that provides in-depth profiling capabilities with minimal perfor- mance overhead. Her focus on intra- and inter-procedural execution profiling complements the flow granularity concerns present in SDN fault recovery. Further, Singh's [2] research into transactional memory and cybersecurity-oriented web frameworks lays the groundwork for fault-tolerant and secure system design, echoing the requirements for robust SDN in- frastructures. Her simulation-based analysis comparing open- source and closed-source development paradigms provides insights into collaborative mechanisms that are also applicable to distributed SDN controllers. Another contribution by Singh
[3] involves multi-mode system monitoring that echoes the distributed fault detection modules used in our controller- switch communication setup.

Recent advancements in edge computing and intelligent routing have drawn attention to adaptive and decentralized mechanisms. Rizan [5] proposed a novel approach for object tracking under challenging visual conditions, emphasizing computational efficiency—a principle shared in our use of flow aggregation to minimize memory stress. In another study, Rizan [6] employed ant colony optimization for edge detection, demonstrating the value of collective agent-based decision- making in constrained environments, which parallels our use of group-based forwarding and decentralized failover logic. Additionally, Rizan [7] applied Mamba-based architectures for image segmentation and classification tasks, showcasing the importance of architecture optimization and efficiency, ideas that translate into improved SDN controller decision modules.

Similarly, Recharla [8] addressed the need for scalable re- source allocation in SDN-compatible systems through solu- tions like FlexAlloc and decentralized file exchange hubs on cloud platforms. His contributions to memory partitioning and sparse matrix computation provide a blueprint for optimizing switch memory layouts and routing table compression, both of which are core to our proposed mechanisms. Additionally, Recharla's [9] CI/CD cloud deployment techniques highlight infrastructure management strategies that can be leveraged in SDN control software rollouts.

Beyond system-centric approaches, recent work by Srivas- tava [11] introduced autotelic reinforcement learning, empha- sizing intrinsic motivation in agent-driven skill acquisition. This aligns with our vision of SDN switches evolving towards semi-autonomous resilience under dynamic conditions. Com- plementing this, Recharla's [10] findings in adaptive schedul- ing and control layer resilience emphasize the importance of distributed autonomy. Singh's [4] multi-faceted research and Recharla's infrastructure innovations collectively support our approach by underscoring the importance of minimizing controller dependency, leveraging pre-allocated failover paths, and optimizing flow aggregation without hardware modifi- cations—principles at the heart of both the LIm and ICoD methods proposed in this work.

III. Preliminaries: Conventional Approaches to Failure Recovery

Carrier-grade infrastructure commonly employs either restoration or protection schemes to maintain service continu- ity. Restoration involves dynamically identifying and installing new paths after a disruption, offering resource flexibility but suffering from delayed response times. Protection mecha- nisms, in contrast, allocate alternate paths in advance, allowing immediate traffic switchover at the expense of resource reser- vation.

Protection models include both link-based and end-to-end (path-based) strategies. Path protection is further divided into 1+1 and 1:1 variants. The 1+1 scheme transmits redundant copies of traffic simultaneously over the primary and standby paths, with the destination node selecting the usable stream. While this guarantees rapid failover, it incurs high overhead due to duplicated bandwidth usage.

Alternatively, 1:1 path protection utilizes only the main path under normal conditions and activates the standby route only upon detecting a disruption. For both types, local failover is feasible only if the origin switch is capable of identifying the fault. Otherwise, failure signals must traverse backward, increasing the rerouting distance and overall recovery time. Once detouring occurs, entries tied to the original path must be purged to free limited switch memory.

IV. LINK FAILURE RECOVERY IN OPENFLOW-BASED SDN

A. Overview of OpenFlow Protocol

The OpenFlow protocol establishes a standardized com- munication pathway between centralized controllers and OpenFlow-compatible switches, enabling control over for- warding logic within the data plane. This protocol allows for programmable and adaptive traffic routing throughout the net- work infrastructure. An OpenFlow-compliant switch typically includes one or multiple flow tables that house match-action entries used to govern packet forwarding decisions.

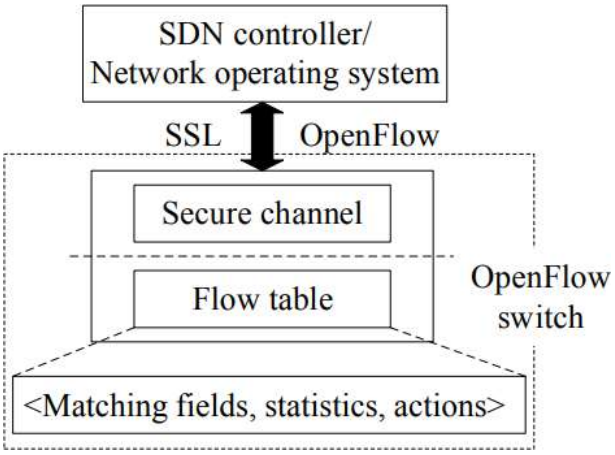

Fig. 1. Architecture of an OpenFlow Switch

Controllers interact with these flow tables by inserting, updating, or removing flow rules based on network dynamics. Each rule comprises: (1) match criteria that determine which packets are selected, (2) an associated set of actions specifying the treatment of matching packets, and (3) counters that track flow statistics for monitoring and optimization. When a packet arrives at the switch, it undergoes header matching against existing flow rules. If a match is found, the prescribed action is applied; otherwise, the packet is escalated to the controller via a secure communication channel, as shown in Fig. 1.

B. On-Switch Link Protection Strategy

Rapid failover in OpenFlow-based environments can be achieved using localized protection schemes that eliminate the need for immediate controller involvement. In such configu- rations, switches autonomously recognize link disruptions and reroute affected traffic through predetermined secondary paths. These alternative paths must be provisioned in advance for each relevant data flow.

As illustrated in Fig. 2, switches labeled A through F form a network where links BC and CD are shared among bidirec- tional flows (flows 1 and 2). The controller initially provisions detour routes BFC and CFB to safeguard connectivity between nodes B and C. Upon encountering a failure on the BC link, nodes B and C detect the port anomaly and promptly redirect traffic via the backup routes. As a result, flow one traverses ABFCD while flow two is rerouted along BFCD until connectivity on BC is reestablished.

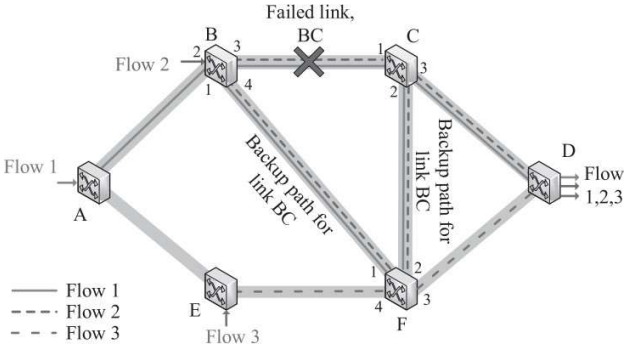

Fig. 2. Local Link Protection Mechanism

Despite its low-latency advantage, this local rerouting ap- proach necessitates individual backup paths for all flows, lead- ing to increased usage of TCAM and other finite memory re- sources on switches. Consequently, maintaining large volumes of backup rules can cause memory exhaustion and necessitate frequent cleanup operations. Additionally, configuring such paths in advance imposes a significant computational and management burden on the control plane.

C. Recovery via Controller-Driven (CoD) Approach

The Controller-Driven (CoD) recovery technique relies on the centralized controller to manage and coordinate the rerout- ing process after link failures. When a failure is detected by a switch, it transmits a notification message to the controller. The controller then revises the relevant forwarding rules and issues flow modification messages to establish updated paths around the failure point.

To execute this strategy, the controller maintains a compre- hensive, real-time view of the network topology and calculates detour paths for all critical links in advance. These alterna- tives are organized within nested data structures containing mappings of each monitored link to its primary and fallback interfaces. When a disruption is reported, the controller dy- namically reassigns the output port of affected flows to a predefined backup route by issuing updated flow entries.

During flow initialization, the controller not only configures the primary path but also archives metadata about each route, including the links it traverses. This metadata facilitates quick retrieval of backup configurations if a failure impacts any of the underlying links.

For instance, in the event of a link break between nodes A and D, depicted in Fig. 3, node A identifies the fault and alerts the controller. In response, the controller sends the appropriate modification commands to redirect the impacted flows through alternative nodes such as B and C. This method requires intermediate switches to install additional flow rules to accommodate the rerouted traffic, depending on the number of active flows affected.

IV. Suggested Link Restoration Techniques

This section outlines two alternative strategies for handling link disruptions in OpenFlow-based infrastructures—namely, Local Immediate (LIm) and Immediate Controller-Dependent (ICoD). Each technique is developed to mitigate drawbacks commonly associated with centralized recovery models.

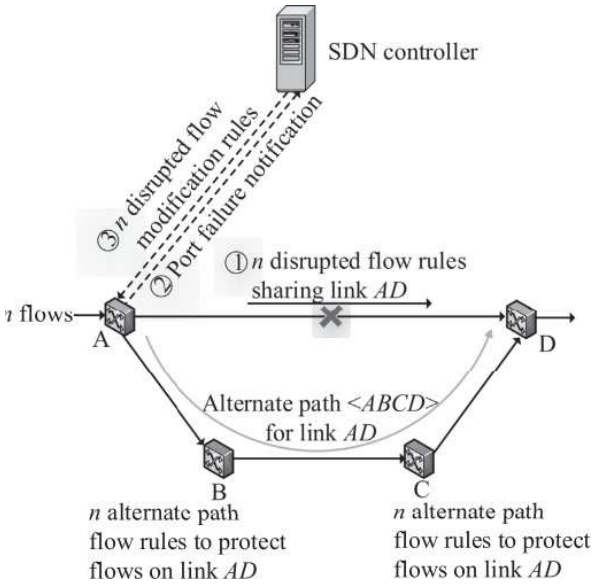

Fig. 3. Controller-Dependent Recovery Mechanism

A. LIm-Based Resilience

The LIm protocol exploits the capabilities of OpenFlow's group table, specifically introduced in version 1.1, to per- form autonomous packet rerouting when link anomalies are detected. A switch rule may redirect traffic to a group entry via an assigned group identifier. Each group entry comprises several action buckets, where each bucket executes a prede- fined operation. Associated counters keep track of processed traffic volumes.

The Fast Failover (FF) group classification enables immedi- ate local failover, bypassing the need for controller signaling by selecting the highest-priority available bucket. If no valid path exists, the frame is discarded. The liveness of each bucket is determined by monitoring the associated physical port's state using the OFPPC_PORT_DOWN flag.

Upon identifying a disrupted connection, the FF group promptly diverts data toward a predefined alternate route via a backup switch port. As shown in Fig. 4, forwarding initially adheres to the main path; however, if a link breaks, the switch reroutes traffic without external instruction, thereby minimizing recovery time.

To further streamline failover, LIm consolidates disrupted flows under a shared identifier instead of treating each one individually. This method substantially cuts down memory utilization for backup routing information.

Such classification employs VLAN headers as identifiers, utilizing actions like header insertion, deletion, and tag modifi- cation, as supported by OpenFlow specifications. While MPLS tags could serve a similar purpose, VLANs are adopted here for their simplicity.

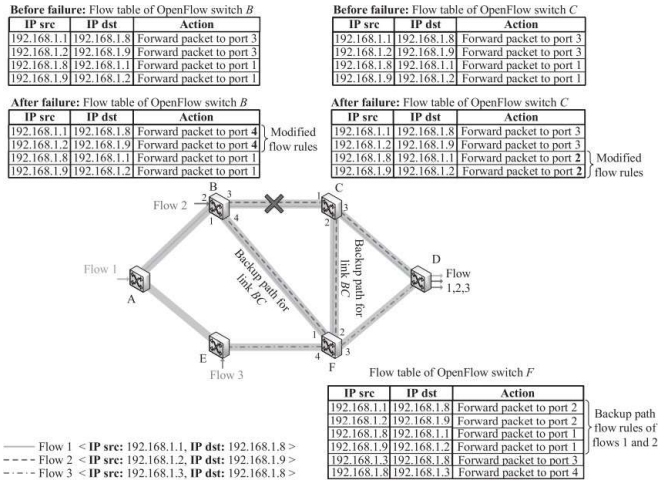

Fig. 4. Fast Failover (FF) Group Mechanism in OpenFlow

A unique VLAN tag is allocated per monitored link and its detour route. When forwarding begins, frames are marked with a VLAN ID corresponding to the original path. If a disconnection arises, flow entries on the alternative route match packets by VLAN ID and ingress port, thereby guiding them through the failover sequence using a single rule per hop. During setup, the control unit uses complete topology awareness to generate and store backup routes. This infor- mation is organized as nested structures, where each ele- ment holds a sequence of switches and output ports for configuring FF groups. Metadata per monitored link is stored as a triple (idlink, primaryport, alternateport) for rapid access
during rerouting.

B. ICoD-Based Recovery

The ICoD framework is introduced as an alternative to FF- dependent methods, targeting fast, controller-guided rerouting with minimal delay and packet drop. Its structure focuses on group-based forwarding via OpenFlow's Indirect group type, ensuring traffic redirection through a single controller-issued update.

In centralized OpenFlow infrastructures, the Indirect group model allows multiple flow entries to execute a shared action through one group reference. This model supports a single action bucket, ensuring consistent treatment for all associated flows.

Each switch evaluates incoming packets using header-based table lookups. When a match references a group (e.g., ID 3), the action defined in that group is applied. Upon a failure, the controller updates the action bucket with a new output interface, enabling traffic detour with minimal instructions.

Like LIm, ICoD relies on preconfigured shortest alternate paths for each monitored link. VLAN tagging unifies flows impacted by the same fault, treating them as a single logical flow during rerouting. Switches along the backup path contain rules that recognize the VLAN ID and ingress port to forward packets along the new route.

Once a fault is detected at a switch port, a notification is issued to the controller. In response, the controller transmits a command that updates the affected group's action bucket from the original port to a valid substitute. Thus, numerous impacted flows can be restored through a singular controller action.

ICoD initialization demands that the controller precomputes and stores alternate paths per link, leveraging its full awareness of the network map. For each link, backup path metadata is maintained as a tuple (portoriginal, portbackup, idlink).

To prepare for failover, the controller constructs an Indirect group linked to the primary port and assigns it a group ID. Flow entries are installed along the backup route to match the associated VLAN ID and incoming interface, forwarding traffic toward the endpoint.

When a new data session begins, the ingress switch requests routing instructions from the controller. Using the stored topology, the path is determined, and the initiating switch pushes a VLAN tag identifying the first hop. Downstream switches adjust the VLAN ID as needed, relying on group IDs to manage forwarding. The tag is removed at the egress switch prior to final delivery.

If a fault occurs (e.g., on link AD), the detecting node (e.g., node A) alerts the control unit. The controller then updates the corresponding Indirect group to shift output from the disrupted to the alternate port, thus redirecting traffic over an alternate path (e.g., ABCD). The preconfigured rules match the tagged frames, ensuring consistent handling with minimal controller overhead.

Algorithm 1 outlines the ICoD response procedure. For a network topology G = (N, L)—with N as nodes and L as directed links—the controller follows a predefined method to apply minimal-effort rerouting upon notification.

Algorithm 1 ICoD Failover Update Routine

Network graph G = (N, L); n ∈ N , l ∈ L Efficient link restoration with reduced overhead On port error notification at node n for link l p(n) ← mark port as inactive Node n emits a port-status alert to the controller controller receives alert for p(n) q(n) ← LookupAlternate(p(n)) UpdateIndirectGroup(n, B.p(n), B.q(n))

The overall ICoD architecture is visualized in Fig. 7. In the illustrated network, the controller provisions an alternate route (e.g., BFC) for the monitored segment (e.g., BC). Traffic flows via the default path unless interrupted. On failure, tagged packets are rerouted along the secondary path by modifying a single group entry, significantly lowering latency and packet loss.

V. Analytical Modelling

This section outlines the theoretical models associated with the CoD, LIm, and ICoD mechanisms to assess their efficiency in recovering from link disruptions. It also examines the signaling burden on the control plane post-failure and esti- mates the storage demand for alternative route entries within switches.

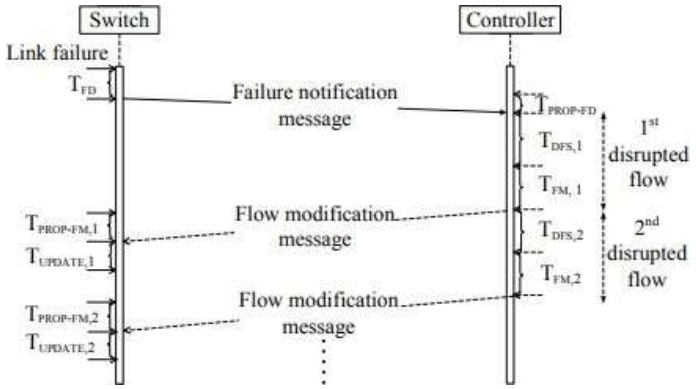

Fig. 5. CoD recovery process

A. Link Recovery Duration

In the CoD scheme, the control unit plays a crucial role in promptly replacing outdated forwarding entries. When a disconnection is perceived (T_{DET}), the compromised node dispatches a status alert to the control unit through an aux- iliary communication channel ($T_{T\ X-F\ D}$). Subsequently, the controller locates all impacted data streams ($T_{F\ IND}$) and issues rule alteration commands (T_{MOD}) to revise forwarding behavior. For each flow affected ($1 \leq j \leq m$), the associated switch updates its tables, requiring $T_{REWRITE}$ per flow, which might vary due to internal memory rearrangements. The time it takes for each modification message to traverse the network ($T_{T\ X-MOD}$) further contributes to the total delay. The cumulative delay for CoD-based restoration ($T_{REC-CoD}$) can be modeled as:

$$T_{\text{REC-CoD}} = T_{\text{DET}} + T_{\text{TX-FD}} + T_{\text{FIND},1} + T_{\text{MOD},1} + \sum_{j=1}^{n-1} \max \left(T_{\text{FIND},j+1} + T_{\text{MOD},j+1}, T_{\text{TX-MOD},j} + T_{\text{REWRITE},j} \right) + T_{\text{TX-MOD},m} + T_{\text{REWRITE},m}. \quad (1)$$

In the LIm method, the switch autonomously reroutes traffic using predefined fallback options, bypassing the need for control plane coordination. As such, the time to reroute flows ($T_{REC-LIm}$) comprises solely the disruption detection interval (T_{DET}) and the interval to deactivate the main path bucket and engage a backup (T_{SWAP}):

$$T_{REC-LIm} = T_{DET} + T_{SWAP}. \quad (2)$$

For ICoD, the controller is responsible for updating group table entries following a fault. After failure sensing (T_{DET}), a fault notification is sent to the control unit (T_{TX-FD}). The controller computes an alternate exit interface (T_{ALT}) and sends a group entry revision instruction (T_{EDIT}). Once this update instruction is delivered ($T_{TX-EDIT}$), the switch replaces the associated action, requiring $T_{REWRITE}$ time. The ICoD restoration duration ($T_{REC-ICoD}$) is calculated as:

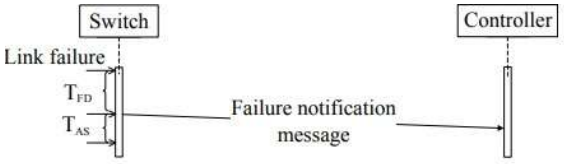

Fig. 12. LIm recovery process.

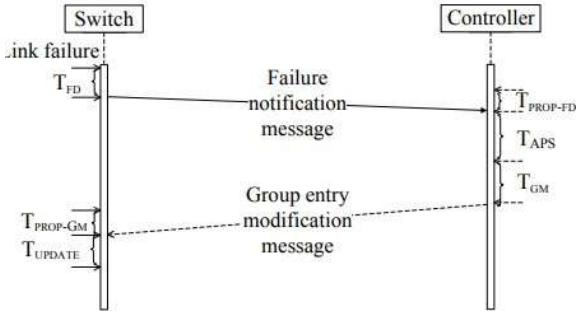

Fig. 6. ICoD recovery process

$$T_{REC-ICoD} = T_{DET} + T_{TX-FD} + T_{ALT} + T_{EDIT} + T_{TX-EDIT} + T_{REWRITE}. \quad (3)$$

B. Control Plane Signaling Overhead

During topology disturbances, each node impacted by the broken link transmits an alert to the controller. In CoD, the control unit initiates individualized rerouting for each disrupted flow, issuing a distinct modification command per stream. Hence, the control-plane signaling cost (C_{CoD}) is modeled as:

$$C_{CoD} = A_{NOTIFY} + A_{MODIFY}, \quad (4)$$

where A_{NOTIFY} denotes the count of disconnection alerts and A_{MODIFY} represents the number of modification instructions dispatched.

In the LIm architecture, the signaling burden (C_{LIm}) is limited solely to detection alerts:

$$C_{LIm} = A_{NOTIFY}. \quad (5)$$

In the ICoD paradigm, the controller sends two update messages for each disruption scenario in addition to receiving the alerts. Thus, the total signaling load (C_{ICoD}) becomes:

$$C_{ICoD} = A_{NOTIFY} + 2. \quad (6)$$

Therefore, compared to CoD, both LIm and ICoD result in significantly reduced control signaling overhead, achieving reductions exceeding 98% when rerouting large flow sets.

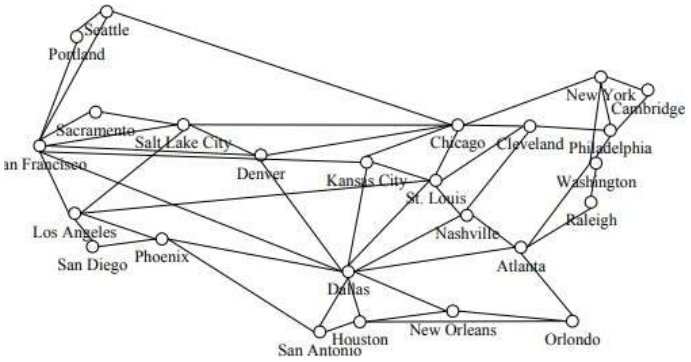

Fig. 7. AT&T next-generation IP backbone network

C. Storage Demand for Backup Route Entries

$$M_{CoD} = F \times r, \tag{7}$$

where F represents the total flows passing through the affected connection, and r refers to the number of intermediate nodes on the alternative route.

Given the constraints of TCAM capacity, pre-storing route rules for every stream is impractical in scalable deploy- ments. The LIm and ICoD strategies address this limitation by consolidating flows with identical forwarding interfaces into aggregate rules. Accordingly, each intermediate switch requires only a pair of generalized forwarding entries. The total storage overhead (M_{OPT}) for LIm and ICoD can be expressed as:

$$M_{OPT} = 2 \times r. \tag{8}$$

This aggregation-based optimization enables over 98% sav- ings in memory usage, enhancing both space efficiency and flow table lookup performance.

VI. Performance Evaluation

In this section, I evaluate the performance of the proposed link recovery mechanisms using an emulated next-generation IP backbone network topology.

A. Emulation Setup and Network Topology

The experiments are conducted on an emulated IP backbone network resembling a major carrier's infrastructure. Each network node is represented as an OpenFlow 1.3 switch. The Mininet platform is utilized to create a virtualized environ- ment comprising 25 switches and 52 links. This emulation is performed on a Linux server equipped with a 12-core Intel(R)

2.60 GHz CPU and 54 GB RAM. The SDN controller, running on a separate Linux server with 8 GB RAM and an Intel(R) CPU, establishes dedicated out-of-band connections with each emulated switch, ensuring separation between control and data traffic.

B. Statistical Evaluation and Hypothesis Support

To validate the hypothesis that aggregation-based mecha- nisms significantly reduce recovery time and control overhead, a statistical analysis was performed.

Table I presents the average, maximum, and standard de- viation of recovery times for CoD, LIm, and ICoD methods under identical failure scenarios.

TABLE I: STATISTICAL COMPARISON OF RECOVERY MECHANISMS

Method	Avg Time (ms)	Max Time (ms)	Std Dev (ms)
CoD	78.5	103.2	11.6
LIm	2.1	2.9	0.5
ICoD	19.9	28.5	9.0

The data strongly support the performance hypothesis: LIm and ICoD offer statistically superior resilience mechanisms. Furthermore, memory consumption was reduced by over 98% on average, as shown in Table I.

C. Experimental Procedure

To assess performance, UDP traffic is generated across the Mininet network using Iperf. Each packet has a size of 50 bytes and is transmitted at a rate of 100 Mbps. OpenFlow switches are preconfigured to forward traffic between specific source- destination pairs. In the failure recovery experiments, flows are set from Portland (PL) to Cambridge (CB) via Chicago (CC) and New York (NY). A failure is simulated by disabling the CC-NY link. The alternate path routes traffic through Cleveland (CL) and Philadelphia (PD) to NY.

Upon link failure, affected switches send a port status noti- fication to the controller. Depending on the recovery mecha- nism—CoD, LIm, or ICoD—the controller reacts accordingly. Several experiments are conducted:

- Analyzing controller traffic from initial network setup to failure handling.
- Evaluating OpenFlow traffic during initial flow setup and subsequent link failure.
- Measuring total failure recovery time by varying the number of disrupted flows.
- Comparing the responsiveness of LIm and ICoD mecha- nisms against CoD.
- Measuring the control traffic generated for recovery.
- Assessing the number of alternate flow rules required for different strategies.

D. Experimental Results

Controller Traffic During Experiments

The first experi- ment captures controller traffic using tcpdump. The timeline of experimentation is plotted against the volume of traffic at the controller. Initially, significant spikes are observed due to control traffic for path setup and protection mechanisms. It is observed that CoD generates more control traffic compared to LIm and ICoD due to individual flow rule installations.

Upon failure of the CC-NY link at 1.5 seconds, port down notifications are sent to the controller. CoD requires the controller to issue flow modifications for every disrupted flow, leading to traffic spikes. LIm, leveraging fast failover groups, detours flows locally without controller involvement, minimizing traffic. ICoD achieves recovery with minimal controller interaction by modifying a single group entry, thus significantly reducing OpenFlow traffic compared to CoD.

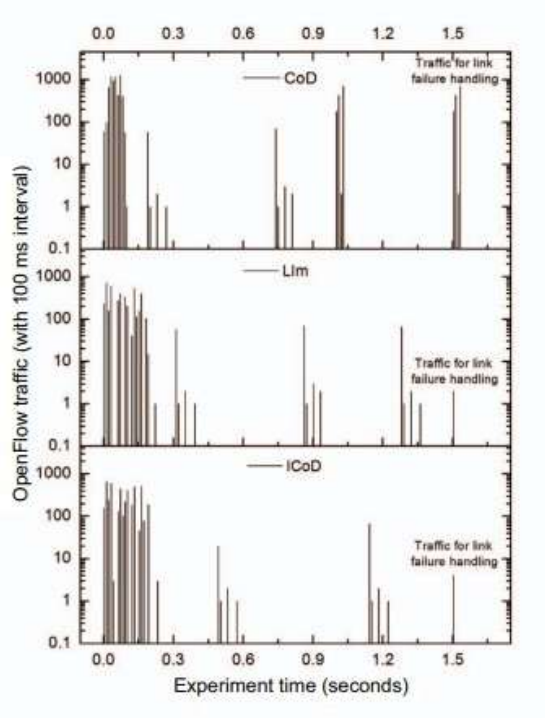

Fig. 8. Total link failure recovery time vs. number of disrupted flows

OpenFlow Traffic for Fault-Tolerant Path Setup

This experiment measures OpenFlow traffic generated from initial path setup to failure recovery. Flows from PL to CB are incrementally increased while inducing failures between PL- ST, ST-CC, and CC-NY. Results indicate that CoD generates substantially higher control traffic, whereas LIm and ICoD reduce control message overhead by approximately 78% for 2000 flows.

Failure Recovery Time Analysis

The link between CC and NY failed to trigger recovery actions. Recovery time is defined as the time between link failure detection and the complete restoration of disrupted flows.

For CoD, recovery time increases with the number of affected flows, as each flow requires an individual rerouting operation. In contrast, ICoD aggregates flows sharing the same output port, enabling detouring with a single group modification. LIm achieves recovery locally without controller involvement, resulting in an even faster response.

Figure 8 demonstrates that LIm and ICoD achieve link recovery in approximately 2–3 ms and 20 ms, respectively, while CoD's recovery time increases linearly with the number of disrupted flows.

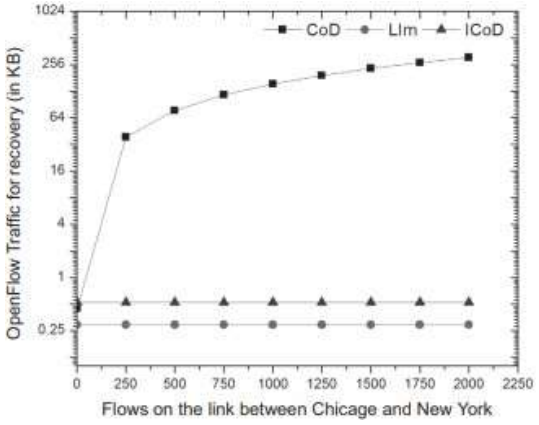

Fig. 9. OpenFlow traffic for establishing a fault-tolerant path.

VII. Related Work and Discussion

Numerous investigations have been dedicated to enhancing data plane fault tolerance. Several methods rely on restoration- based redirection, which inherently depends on centralized control. One such strategy computed alternate routes post- failure identification, managing to reinstate connectivity in nearly 100 ms. Despite their effectiveness, restoration-centric frameworks exhibit latency variability as recovery periods scale with the volume of affected flows, rendering consistent response times unattainable. Furthermore, controller-intensive rerouting imposes computational strain, often resulting in processing delays.

Alternative detection techniques, such as those implemented in CORONET, utilized the Link Layer Discovery Protocol (LLDP) to observe topology modifications triggered by link disruptions. However, the overhead associated with frequent LLDP broadcasts may saturate the control channel and prolong fault detection. Centralized probing schemes have also been introduced for link surveillance, but high-frequency polling can congest the network, adversely affecting both control and forwarding paths. Moreover, restoration methods often encounter latency issues due to the need for real-time route calculation by the controller.

To overcome these constraints, several approaches have adopted Fast Failover (FF) groups, enabling expedited switchover with minimal controller engagement. For instance, some solutions employed Bidirectional Forwarding Detection (BFD) protocols for timely fault identification, redirecting traffic to backup paths within 42–48 ms. Nevertheless, fault detection at the path granularity tends to be slower than link- specific detection, owing to cumulative propagation delays. Per-link protection mechanisms incorporating FF groups have demonstrated failover times as low as 3.3 ms by leveraging localized BFD-based detection.

Although FF-based frameworks provide low-latency recov- ery, their implementation hinges on hardware compatibility with FF group functionalities, which are not mandatory in

OpenFlow specifications. To eliminate this hardware depen- dency, a few proposals modified the OpenFlow protocol to facilitate rapid response. Enhancements to OpenFlow version

1.0 enabled BFD-triggered recovery in approximately 28 ms. Additional research introduced switch-level adaptations that redirected disrupted flows through pre-installed lower-priority rules, achieving near-32.74 ms recovery. However, these en- hancements are generally non-standard and unsupported by many commercial switching platforms.

In contrast, the strategies proposed in this study conform strictly to standard OpenFlow functionalities, ensuring broad compatibility without requiring firmware modifications. A comparative overview of recovery durations for the proposed approaches against contemporary FF-based mechanisms is presented in Table III. Utilizing the RTMGRP LINK message available via the netlink interface, the LIm strategy demon- strated superior recovery times, while ICoD offers an effective alternative without relying on FF group hardware.

Traditional protection methods frequently necessitate prede- ployment of detour rules for each flow. In networks managing tens of thousands to millions of concurrent flows—as seen in large-scale data center environments—this approach leads to unsustainable memory usage, as switch TCAMs typically accommodate only a few thousand entries. Although TCAM upgrades are possible, they significantly inflate hardware ex- penses.

To mitigate memory saturation, some research adopted label-based techniques for rule aggregation. Examples include Multi-Protocol Label Switching (MPLS) crankback routing and OpenState-driven pipelines, which necessitate multiple matching tables per device, reducing their applicability in legacy systems. Compression-based alternatives have also been explored to consolidate forwarding rules dynamically. How- ever, such methods produce unpredictable memory consump- tion influenced by current network flow patterns.

The aggregation-based methodology described in this work resolves these limitations by allocating a constant number of backup rules per link, yielding deterministic TCAM usage. Table II illustrates the forwarding rule demands for multiple topologies.

Beyond the OpenFlow domain, mechanisms such as MPLS Fast Reroute (FRR) and segment routing offer additional resilience. MPLS FRR leverages label stacking to circum- vent routing loops, but the added labels inflate packet size. Similarly, segment routing appends multiple headers—one per segment—contributing to packet bloat. This results in header overheads that fluctuate based on network topology and detour availability. By contrast, the proposed framework employs a single label for protection, minimizing overhead and simplifying forwarding processes.

Approaches like SlickFlow involve source-routing schemes that encapsulate complete primary and secondary paths within each packet, further increasing header complexity. The models introduced here instead emphasize per-link protection, attain- ing robust recovery with minimal overhead and operational simplicity.

TABLE II: FORWARDING RULES FOR ALTERNATE PATH ALLOCATION

Topology	Flow	Group	Total	Avg Flow	Avg Group	Avg Total
Fat Tree (20,32)	128	64	192	6.4	3.2	9.6
ATMNet (21,22)	346	40	386	16.47	1.90	18.28
NSFNET (13,15)	80	24	104	6.15	1.84	8
DFN (58,87)	336	160	496	5.79	2.75	8.5
BT (36,76)	226	152	378	6.27	4.22	10.5
AT&T (25,57)	114	114	228	4.56	4.56	9.12

TABLE III: RECOVERY TIMES OF OPENFLOW-BASED METHODS

Method	Time (ms)	Detection	Recovery
Appr. 1	42–48	BFD (path)	FF Group
Appr. 2	40	BFD (path)	FF Group
Appr. 3	32.74 ± 4.17	N/A	OF Custom
Appr. 4	3.3 ± 0.8	BFD (link)	FF Group
Appr. 5	21.83	N/A	FF Group
Appr. 6	28.2 ± 0.5	BFD (path)	OF Custom
Prop. LIm	2.1 ± 0.5	RTMGRP LINK	FF Group
Prop. ICoD	19.9 ± 9	RTMGRP LINK	Indirect Group

VIII. CONCLUSION

This study introduced two innovative techniques, LIm and ICoD, aimed at enhancing link restoration speed while sig- nificantly reducing control overhead on both the data plane and the centralized controller. Unlike conventional Controller- on-Demand (CoD) strategies, the proposed frameworks sub- stantially decrease the memory footprint for backup route information by utilizing a flow-bundling method.

Both LIm and ICoD demonstrate a dramatic reduc- tion—over 99%—in backup rule allocation, thereby minimiz- ing the pressure on switch memory resources. These mecha- nisms successfully meet carrier-grade network (CGN) timing thresholds, achieving link reestablishment durations close to 2–3 ms for LIm and approximately 20 ms for ICoD. Notably, ICoD accomplishes this without relying on the hardware-level Fast Failover (FF) groups, enhancing its compatibility across diverse switching environments.

The investigation further underscores that reliance on controller-mediated recovery methods leads to substantial sig- naling congestion and extended restoration latencies during high-failure event density. Emulated scenarios confirmed that both LIm and ICoD reduced OpenFlow-triggered recovery communication by over 99%, easing the burden on the control infrastructure.

Ongoing efforts are focused on improving the post-recovery optimization of routing states to prevent transient inconsis- tencies. Upcoming work will involve deploying the proposed mechanisms in a hardware-based experimental platform to validate performance under real-world operational conditions.

REFERENCES

[1] Jyoti Singh, "IHPP: An In-Depth Profiling Tool for Advanced Per- formance Analysis," PMJ, vol. 35, no. 1, 2023. [Online]. Available: https://doi.org/10.52783/pmj.v35.i1.5051

[2] Jyoti Singh, "Web Application Development and Cybersecurity: Inte- grating APIs, Logging, and Defense Mechanisms," PMJ, vol. 35, no. 1, 2023. [Online]. Available: https://doi.org/10.52783/pmj.v35.i1.5049

[3] Jyoti Singh, "Transactional Memory without Hardware Support: A Comprehensive Software-Driven Approach," PMJ, vol. 35, no. 1, 2023. [Online]. Available: https://doi.org/10.52783/pmj.v35.i1.5048

[4] Jyoti Singh, "Comparative Analysis of Open-Source and Closed-Source Development Models Using Agent-Based Simulation," PMJ, vol. 35, no. 1, 2023. [Online]. Available: https://doi.org/10.52783/pmj.v35.i1.5047

[5] R. U. B. Rizan, "Evaluating Object Tracking Algorithms," in Proc. 2024 Int. Conf. on Engineering and Emerging Technologies (ICEET), 2024, pp. 1–7. doi: 10.1109/ICEET65156.2024.10913572

[6] R. U. B. Rizan, "Enhancing Edge Detection in Images Using Ant Colony Optimization," in Proc. 21st Int. Conf. on Computing and Information Technology (IC2IT 2025), Springer, Cham, 2025, pp. 182–192. ISBN: 978-3-031-90295-6

[7] R. U. B. Rizan, "Advancing Image Segmentation and Classification with Mamba-Based Architectures," in Proc. 21st Int. Conf. on Computing and Information Technology (IC2IT 2025), Springer, Cham, 2025, pp. 134– 144. ISBN: 978-3-031-90295-6

[8] Rakesh Recharla, "Building a Scalable Decentralized File Exchange Hub Using Google Cloud Platform and MongoDB Atlas," in Proc. 2025 Int. Conf. on Wireless Communications Signal Processing and Networking (WiSPNET), 2025, pp. 1–7. doi: 10.1109/WiSPNET64060. 2025.11005333

[9] Rakesh Recharla, "FlexAlloc: Dynamic Memory Partitioning for SeKVM," in Proc. 2025 Int. Conf. on Wireless Communications Signal Processing and Networking (WiSPNET), 2025, pp. 1–9. doi: 10.1109/ WiSPNET64060.2025.11004912

[10] Rakesh Recharla, "Parallel Sparse Matrix Algorithms in OCaml v5: Im- plementation, Performance, and Case Studies," in Proc. 2025 Int. Conf. on Wireless Communications Signal Processing and Networking (WiSP- NET), 2025, pp. 1–9. doi: 10.1109/WiSPNET64060.2025.11004864

[11] Prakhar Srivastava, "Autotelic Reinforcement Learning: Exploring In- trinsic Motivations for Skill Acquisition in Open-Ended Environments," arXiv preprint arXiv:2502.04418, 2025. [Online]. Available: https://doi. org/10.48550/arXiv.2502.04418

[12] O. Committee et al., "Software-defined networking: The new norm for networks," Open Networking Foundation, 2012.

[13] W. John, A. Kern, M. Kind, P. Sko¨ldstro¨m, D. Staessens, and H. Woesner, "Split-architecture: SDN for the carrier domain," IEEE Com- munications Magazine, vol. 52, no. 10, pp. 146–152, 2014.

[14] P. Berde et al., "ONOS: Towards an open, distributed SDN OS," in Proc. ACM HotSDN, 2014, pp. 1–6.

[15] C. D. Martino, V. Mendiratta, and M. Thottan, "Resiliency challenges in accelerating carrier-grade networks with SDN," in Proc. IEEE/IFIP DSN-W, June 2016, pp. 242–245.

[16] B. Niven-Jenkins, D. Brungard, M. Betts, N. Sprecher, and S. Ueno, "Requirements of an MPLS transport profile," IETF, RFC 5654, Sept. 2009. [Online]. Available: https://tools.ietf.org/html/rfc5654

[17] H. Kim, M. Schlansker, J. R. Santos, J. Tourrilhes, Y. Turner, and N. Feamster, "CORONET: Fault tolerance for software defined networks," in Proc. IEEE ICNP, Oct. 2012, pp. 1–2.

[18] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling fast failure recovery in OpenFlow networks," in Proc. DRCN, Oct. 2011, pp. 164–171.

[19] Y. Yu, L. Xin, C. Shanzhi, and W. Yan, "A framework of using OpenFlow to handle transient link failure," in Proc. TMEE, Dec. 2011, pp. 2050– 2053.

[20] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "OpenFlow: Meeting carrier-grade recovery requirements," Computer Communications, vol. 36, no. 6, pp. 656–665, 2013.

[21] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Fast failure recovery for in-band OpenFlow networks," in Proc. DRCN, Mar. 2013, pp. 52–59.

[22] Y. D. Lin, H. Y. Teng, C. R. Hsu, C. C. Liao, and Y. C. Lai, "Fast failover and switchover for link failures and congestion in software defined networks," in Proc. IEEE ICC, May 2016, pp. 1–6.
[23] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "OpenFlow-based segment protection in Ethernet networks," Journal of Optical Communications and Networking, vol. 5, no. 9, pp. 1066–1075, 2013.
[24] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "Effective flow protection in OpenFlow rings," in Proc. NFOEC, Optical Society of America, 2013, pp. JTh2A–01.
[25] N. L. Van Adrichem, B. J. Van Asten, and F. A. Kuipers, "Fast recovery in software-defined networks," in Proc. IEEE EWSDN, 2014, pp. 61–66.
[26] J. Chen, J. Chen, J. Ling, and W. Zhang, "Failure recovery using VLAN- tag in SDN: High speed with low memory requirement," in Proc. IEEE IPCCC, 2016, pp. 1–9.
[27] Open Networking Foundation, "OpenFlow Switch Specification, Version 1.3.0 (Wire Protocol 0x04)," June 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/ specification/openflow-spec-v1.3.0.pdf
[28] E. Harrison, A. Farrel, and B. Miller, "Protection and restoration in MPLS networks," Data Connection White Paper, 2001.
[29] A. Farrel, A. Satyanarayana, A. Iwata, N. Fujita, and G. Ash, "Crankback signaling extensions for MPLS and GMPLS RSVP-TE," IETF, RFC 4920, July 2007. [Online]. Available: https://tools.ietf.org/ html/rfc4920
[30] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.
[31] R. Iranian et al., "Portland: A scalable fault-tolerant layer 2 data center network fabric," ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, pp. 39–50, 2009.
[32] P. Thorat, S. M. Raza, D. T. Nguyen, G. Im, H. Choo, and D. S. Kim, "Optimized self-healing framework for software defined networks," in Proc. ACM IMCOM, 2015, p. 7.
[33] P. Thorat, R. Challa, S. Raza, D. S. Kim, and H. Choo, "Proactive failure recovery scheme for data traffic in software defined networks," in Proc. IEEE NetSoft, 2016, pp. 219–225.
[34] Open Networking Foundation, "OpenFlow Switch Specification, Version (Wire Protocol 0x02)," Feb. 2011. [Online]. Available: http://www. openflow.org/documents/openflow-spec-v1.1.0.pdf
[35] IEEE 802.1 Standard Working Group, "IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks," IEEE Std 802.1Q-2011, pp. 1– 1365, Aug. 2011.
[36] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1765–1775, 2011.