CDRA: A Community Detection based Routing Algorithm for Link Failure Recovery in Software Defined Networks

Muhammad Yunis Daha[1], Mohd Soperi Mohd Zahid[2], Babangida Isyaku[3], Abdussalam Ahmed Alashhab[4]
Department of Computer and Information Sciences,
Universiti Teknologi PETRONAS, Seri Iskandar, Malaysia[1,2,4]
Department of Mathematics and Computer Science,
Sule Lamido University, Kafin Hausa, Nigeria[3]

***Abstract*—The increase in size and complexity of the Internet has led to the introduction of Software Defined Networking (SDN). SDN is a new networking paradigm that breaks the limitations of traditional IP networks and upgrades the current network infrastructures. However, like traditional IP networks, network failures may also occur in SDN. Multiple research studies have discussed this problem by using a variety of techniques. Among them is the use of the community detection method is one of the failure recovery technique for SDN. However, this technique have not considered the specific problem of multiple link multi-community failure and inter-community link failure scenarios. This paper presents a community detection-based routing algorithm (CDRA) for link failure recovery in SDN. The proposed CDRA scheme is efficient to deal with single link intra-community failure scenarios and multiple link multi-community failure scenarios and is also able to handle the inter-community link failure scenarios in SDN. Extensive simulations are performed to evaluate the performance of the proposed CDRA scheme. The simulation results depicts that the proposed CDRA scheme have better simulations results and reduce average round trip time by 35.73%, avg data packet loss by 1.26% and average end to end delay 49.3% than the Dijkstra based general recovery algorithm and also can be used on a large scale network platform.**

Keywords*—*Software Defined Network (SDN); community detection methods; CDRA; link failure

I. Introduction

The complexity, uncontrollability, and the increasing demand of the Internet has led to low utilization of network resources [1]. To address this problem, a new concept of Software Defined Network (SDN) technology has been introduced. The SDN technology decouples the control plane from the data plane and makes the IP networks programmable [2]. However, like traditional IP networks, SDN technology may also have network failure problems. Unlike other network failure problems, link failure is considered to be the most prevalent network failure in both traditional and SDN networks [3]. Link failures must be rectified as soon as possible, as they can cause network congestion and decrease network efficiency. However, unlike traditional IP networks, the SDN can heal the link failure issue by setting up the controller to shift to another alternate shortest path assigned with OpenFlow [4], [5]. Recovery from link failure can be accomplished either through the proactive failure recovery approach or through the reactive failure recovery approach. In the proactive failure recovery method, backup resources are pre-reserved before the occurrence of the failure scenario. On the other hand, in the reactive failure recovery method, backup resources are reserved after the failure scenario [2]. Both the proactive and the reactive failure recovery approaches have their own benefits and limitations. Based on proactive and reactive failure recovery approaches a variety of research studies have been conducted to make the link failure recovery process more fast and efficient. Among the existing work, the use of a community detection scheme for the failure recovery process of SDN is one of the reactive approaches. The existing work [6], [7] shown good performance, however, they have not addressed the multiple link failure and inter-community link failure problem.

This paper presents a reactive failure recovery approach based CDRA scheme for link failure recovery in SDN. The proposed CDRA scheme is efficient and capable to address the single link intra-community failure scenario and multiple link multi-community failure scenario and also able to handle the inter-community link failure scenarios in SDN. This research study has the following primary contribution:

- The CDRA scheme is proposed which deals with both single and multiple link failure scenarios.
- This CDRA scheme is efficient to deal with intra-community, inter-community and multi-community link failure scenarios.
- The CDRA scheme implemented the Louvain and Infomap community detection methods along with the previous study Girvan and Newman method.

- Lastly this study presents a comparative analysis of all three community detection approaches in-term of their performance after failure scenarios in SDN.

The rest of the paper is organized as follows. Section II will discuss the literature review part. Section III talks about the community detection methods used for the proposed CDRA scheme. Section IV discussed the Proposed CDRA scheme for single and multiple link community and inter-community link failure scenarios in SDN. Section V and Section VI present the mathematical explanation of the proposed CDRA scheme and also discuss an example case study respectively. Simulation setup and result discussion are presented in Section VII. Finally, Section VIII delivers an overall conclusion and future direction based on the result analysis and discussion.

II. Literature Review

Based on the link failure scenarios in SDN, failure recovery approaches are classified into two categories, the first one is the proactive failure recovery approach and the second one is the reactive failure recovery approach [2], [8]. The processing time of proactive failure recovery is fast however, in proactive mechanism, flow entries are installed along with the TCAM (ternary content-addressable memory) which is limited in size, expensive and power-hungry [9], [10].

However, on the other hand, the reactive recovery mechanism is cheaper than the proactive mechanism. But, the reactive failure recovery approach has the latency problem as the network controller will first need to calculate an alternative path and then install the flow entries in the relevant switches [3].

Many researchers have discussed their works by using the proactive and reactive failure recovery approaches to overcome the failure scenarios in SDN. This section presents some of the research studies relevant to the proposed CDRA scheme conducted on the basis of the reactive failure recovery approach for link failure recovery scenarios in SDN.

Sharma et al. [11] describe the reactive failure recovery mechanism in which the controller adjusts the topology and determines a new path for each failure-affected flow after failure detection. After pushing new flow entries and deleting original functioning path flow entries, the redirection will be complete. The results demonstrate that the reactive recovery strategy has larger recovery latency, making it impossible to achieve the carrier-grade requirement. The author [12] demonstrates the restoration based link failure scenario in SDN but the conducted study is performed over small network topology and only discuss the performance of link failure scenarios in SDN. The authors of [13] developed a novel technique for rapid restoration by lowering the processing time, which is normally paid by the controller, by identifying an alternate path (from end-to-end) with a low operation demand. One major flaw in this research is that it does not ensure that the health nodes in the impacted path will be in the same sequence on the alternate path. Furthermore, there is a scarcity of information on the simulation technology that was utilized.

Research work proposed in [14] demonstrated how a quick restoration may be achieved, however, the experimental topologies in both studies were small scale. In addition, the processing time for setting up the selected path was neglected, which is a need in SDNs to re-route traffic from the impacted primary path to the backup way.

Author in [15] suggested a reactive link failure recovery approach based on the shortest path first algorithm. Each path's packets are classified as high or low priority. For high-priority packets, the suggested approach assures the shortest possible delay. The method, however, can only be used on a small-scale network and is not suitable for large-scale SDNs. By spreading traffic evenly among the available paths, the method also reduces congestion. As a result, as the network grows larger, the algorithm's complexity grows. Another flaw in the suggested solution is that the implementation mechanism is given insufficient information. Furthermore, the method has not been validated using conventional internet topology datasets.

Unlike previous studies, the approach developed by [6], [7] used the community detection technique for failure recovery in SDN. In their study, the authors first divide the whole network into cliques. When a link failure occurs, instead of correcting the whole network from the beginning, it is possible to correct only the path in that clique. Thus, the path recovery works faster. But on the other hand, they have not considered the specific problem of multiple link failure and inter-community link failure problem scenarios in SDN and their used community detection method is relatively slow and time costing [16].

In comparison with the previous research studies, this paper proposed a community detection-based routing algorithm for link failure recovery in SDN. The proposed CDRA scheme is capable to deals with both single and multiple link failure scenarios. Moreover, the proposed CDRA scheme is efficient to deal with both the inter and the intra-community failure problem as well. Unlike previous studies, this paper implies Louvain and Infomap community detection methods. Both community detection algorithms accuracy is comparable to other community detection algorithms and also have better scalability [17]–[19]. A summary of the related work to reactive failure recovery relevant to the proposed CDRA scheme is presented in Table I.

III. Community Detection Methods

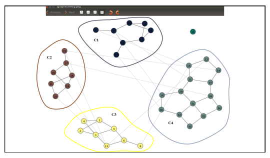
The community detection methods can increase the speed and efficiency with which networked data is processed, analyzed, and stored. Various types of network-related problems, such as routing and data forwarding, have been solved using community detection [20]

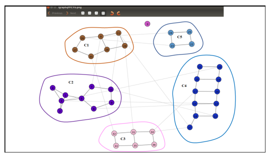
TABLE I. Summary of the Related Work

Author	Year	Recovery Approach	Link Failure Type	Community Detection Method	Limitations
Sharma [11]	2011	Reactive	Single Link Failure	No	Large Recovery Time Delay
Sharma [14]	2013	Reactive and proactive	Single Link Failure	No	The experimental topologies were small scale. In addition, the processing time for setting up the selected path was neglected.
Astaneh [13]	2016	Reactive	Multiple Link Failure	No	Does not ensure the health nodes in the impacted path, and there is a scarcity of information on the simulation technology.
Malik [6]	2017	Reactive	Single Link Failure	Girvan and Newman	Not consider addressing Inter community failure and multiple failure.
Muthuma- [15]	2017	Reactive	Single and Multiple Link Failure	No	Can only be used on a small-scale network. Secondly, When network grows larger, the algorithm's complexity grows.
Malik [7]	2020	Reactive	Single Link Failure	Girvan and Newman	Not consider Inter community failure and multiple failure. Used relatively Slow community detection Method.
Yunis [12]	2021	Reactive	Single&multi Link Failure	No	Studied link failure performance over small network topology in SDN.

[21]. Many algorithms have been developed to recognise communities like structures in networks such as Girvan and Newman, 2002 [22], Clauset, 2004 [23], Louvain, 2008, [24] Infomap, 2008 [25]. This section presents a comparative analysis between Louvain, Infomap, and the Girvan and Newman community detection algorithms. Fig. 1 shows the network communities obtained from Cost266 network topology [26] by implementing the Louvain, Infomap and Girvan and Newman community detections methods.

(a) Cost266 Network Topology


(b) Cost266 with Girvan and Newman Community Method


(c) Cost266 with Louvain Community Method

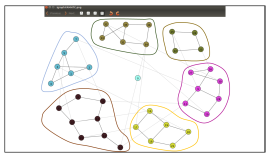

(d) Cost266 with Infomap Community Method

Fig. 1. Implementation of Community Detection Methods over Cost266 Network Topology.

In a previous study, [6], [7] author, used the Girvan and Newman community detection approach for partitioning the network graph into different network communities. The Girvan and Newman community detection algorithm identifies edges that lie between communities in a network and removes them thereby allowing for the identification of distinct communities in the network.

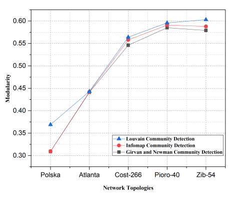
However, on the other hand, when dealing with a large network graph, this algorithm is not particularly efficient and not recommended for large data sets. Moreover, the Girvan and Newman algorithm has time complexity $\mathcal{O}(m^2n)$ which make this algorithm relatively slow and time costing as compared to the Louvain and Infomap community detection method [16], [24], [27], [28].

Unlike the previous study, based on the concept of modularity, this paper used the Louvain and Infomap community detection method along with the previous study community detection method for partitioning the network graph. High modularity networks feature extensive connections between nodes inside communities, but sparse connections between nodes in different communities [16]. The Louvain community detection algorithm is a hierarchical technique that combines communities into one node repeatedly and performs the modularity clustering on the condensed network. The Louvain algorithm efficiently detects large-scale communities and achieves high modularity. This optimizes for each community the modularity score, where the modularity quantifies the quality of node assignment to the groups. The time complexity of Louvain community detection method is $\mathcal{O}(n \log n)$ [16], [24].

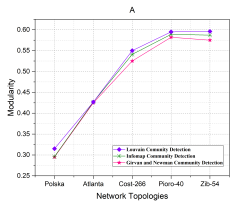
Similar to the Louvain community detection method, the Infomap is another community detection method that allows for the creation of high-quality communities. Infomap method figures out communities by employing random walks to analyze the information flow through a network. The smaller the number of candidates, the more information about the original network has been transferred. Furthermore, it also tries to minimize the cost function of a network graph. The Infomap community detection method runs on time complexity of $\mathcal{O}(m)$ which make this method more efficient [16], [25].

Both algorithm's accuracy is comparable to other community detection algorithms and also have better scalability [17]–[19].

Fig. 2 represents that the modularity obtained by the Louvain and Infomapof community detection methods of different network topologies before and after failure scenarios are higher than the modularity obtained by Girvan and Newman community detection method.



(a) Modularity before Link Failure Scenario.



(b) Modularity after Link Failure Scenario.

Fig. 2. Modularity of Different Network Communities for Different Community Detection Methods.

IV. CDRA: A Proposed Failure Recovery Scheme in SDN

Based on this principle of community detection algorithms, a community detection-based routing algorithm is proposed. The proposed CDRA scheme efficiently deals with both single and multiple link failure scenarios. The proposed CDRA scheme divides a network graph into a small number of different network communities by implementing the community detection algorithms as mentioned in the previous section. Once the network graph is split into different small communities, a data packet is sent from the source node to the destination node. Data packet passes through different network communities that belong to that primary path.

During this data transmission process, once failure happened, most probably either inside the single community or inside two different communities or even between two different communities belonging to that affected path, then instead of searching and correcting throughout the entire graph the proposed CDRA scheme only considered the failure affected communities for treatment. Through this proposed CDRA scheme, only the failure-affected communities are required to install a new path, and the rest of the communities are not disturbed and carry on their duties. This helps the controller to directly deal with that particular community switches and assigned new flow entries instead of searching through the entire network graph. Partitioning the network graph through community detection approaches and directly dealing only with failure affected network communities instead of treating the entire network graph, makes the CDRA scheme more efficient and productive as compared to the general recovery algorithm which considers the complete graph in the failure recovery scenario.

Fig. 3, represents the research framework of the proposed CDRA scheme which is composed of SDN controller, topology analyzer, community producer and route finder. The POX controller used as a remote controller that communicate with application plane and data plane through Northbound API and Southbound API respectively. The topology analyzer is used to analyze the network topology graph through POX OpenFlow-discovery. The community producer and route finder, are the two main components of this research framework and our contribution lies in these two components.

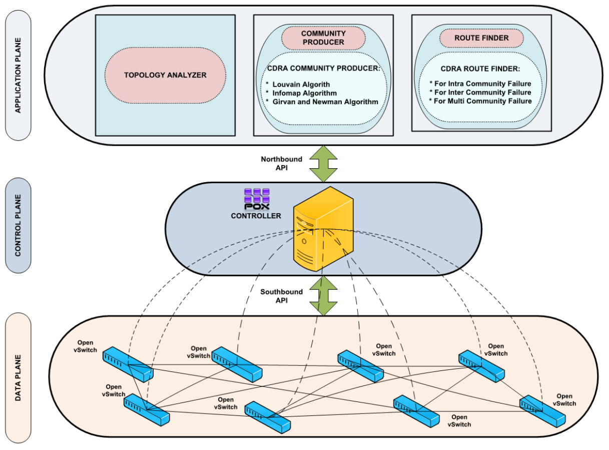


Fig. 3. Research Framework for the Proposed CDRA Scheme.

A. Proposed CDRA Scheme for Community Producer and Reroute Finder

The proposed CDRA scheme implies the community detection methods to determine link failure recovery in SDN. The CDRA scheme can be classified into the following two steps:

The first step of the proposed CDRA scheme is the community producer. The community producer split the network topology graph into different sizes of small network communities by using the Louvain, Infomap, and Girvan and Newman community detection methods. The obtained network communities after implementing the community detection methods can have a different number of nodes and links. Different colors are used for different network communities to make them easy identifiable. The second step of the proposed CDRA scheme is to determine the shortest path from the source node to the destination node passing through different network communities before and after the occurrence of link failure scenarios. Two algorithms have been developed for the second step of the router finder.

The first algorithm is known as the Dijkstra based general recovery algorithm. The general recovery algorithm is based on the simple Dijkstra algorithm that describes the default action which is done through the SDN controller when link failure happens. Algorithm 1 eliminates the flow entries of failure effected path and then install the rules for back-up path from source node to destination node after the occurrence of link failure scenarios. Algorithm 1, finds the shortest path without using the community detection method. The pseudocode of this general recovery algorithm is presented in Algorithm 1.

Algorithm 1 Dijkstra based general recovery algorithm to find the shortest path from network graph N_G

```
1: Input: Implementing the Dijkstra based Recovery Algorithm for Network Graph:= $N_G$
2: Output: Getting Shortest path based on the Dijkstra based Recovery Algorithm
3: Start
4: Setting-up the Dijkstra based Recovery Algorithm for Network Graph:= $N_G$
5: When Link is up:
6: Set Primary path as short path
7: $P_p \in P_{short}\ (S_n,\ D_n)$
8: When Link is down:
9: Set Secondary path as short path
10: $S_p \in P_{short}\ (S_n,\ D_n)$
11: $P_{short}\ (S_n,\ D_n)$ := $P_{short}\ (S_n,\ D_n)$ - $P_p$
12: $S_p$ := $D_g[P_{short}\ (S_n,\ D_n)]$
13: End
```

Algorithm 2 CDRA: Community based routing algorithm for network graph N_G

```
1: Input: Implementing the CDRA approach for Network Graph:= $N_G$
2: Output: Getting Shortest path based on the CDRA approach
3: Start
4: Setting-up the CDRA approach for Network Graph:= $N_G$
5: When Link is up:
6: Set Primary path as short path
7: $P_p \in P_{short}\ \{(S_n, N_{c_S}),\ (D_n, N_{c_D})\}$
8: When Link is down:
9: Set Secondary path as short path
10: $S_p \in P_{short}$
11: $P_{short}\ \{(S_n, N_{c_S}),\ (D_n, N_{c_D})\}$
12: $P_{short}\ \{(S_n, N_{c_S}),\ (D_n, N_{c_D})\}$ - $P_p$
13: $S_p$ := $D_{N_c}[P_{short}\ \{(S_n, N_{c_S}),\ (D_n, N_{c_D})\}]$
14: End
```

The second algorithm is known as the proposed community detection-based routing algorithm. The proposed CDRA scheme is discussed in Algorithm 2. Algorithm 2 is compared with the Dijkstra based general recovery algorithm 1 and set-up it as a meaningful and appropriate benchmark based on the previous research study [6], [7]. Algorithm 2 shows the proposed CDRA scheme that find the shortest path from source node to destination node of a network graph before and after the occurrence of link failure scenarios. Algorithm 2 implies the community detection methods for finding the shortest path after the occurrence of link failure scenarios. The contribution of the proposed CDRA scheme lies in Algorithm 2. The pseudocode of the proposed CDRA scheme is presented in Algorithm 2.

V. Mathematical Explanation of Community Detection Method for the Proposed CDRA Scheme

Mathematically a network graph G is a combinational set of vertices "V" and edges "E" as G= (V, E). Both vertices and edges connect different set nodes in the network graph. By implementing the community detection approach a network graph can be split into the different numbers of communities and every network community defined the subset of the network graph as $N_c \subseteq$ G. Where the network community is the combination set small network communities $N_c = (N_{c_1}, N_{c_2}, ...N_{c_n})$. Every small network community is also the combination of different set of nodes and links presented in equation 1.

$$N_{c_1} = (v_1, e_1)|v_1 \subseteq V \wedge e_1 \subseteq E \quad (1)$$

A path "P" is a set of distance that start from a source node "Sn" of a network community and ends at the destination node "Dn" of an other network community, passes through different set of different small network communities as presented in equation 2.

$$P = \{S_{n_1}, N_{c_1}, S_{n_{1+n}}, N_{c_{1+n}},,, D_n, N_{c_d}\} \quad (2)$$

Once the path "P" is set up between a source node and the destination node the concept of link failure is introduced and failure recovery scenarios are presented as follows:
The first failure recovery scenario represents the single link intra community failure scenario, when link link is down between two node which belongs to the same network community N_{c_1}.

Equation 3, define the first failure recovery scenario, when failed link between two different node $(S_{n_1}, N_{c_1}, S_{n_{1+n}}, N_{c_1})$ belongs to to the same network community N_{c_1}.

$$F_{N_{c_1}} = (S_{n_1}, N_{c_1}, S_{n_{1+n}}, N_{c_1})|\exists(N_{c_1} : N_{c_1}) = (v_1, e_1) \subseteq F_{N_{c_1}} \in e_1) \quad (3)$$

Unlike, first failure recovery scenario, the second failure recovery scenario is about the inter community link failure where the failed link belongs between two different nodes present inside two different network communities.

Equation 4, shows the inter community link failure scenario between node (S_{n_1}, N_{c_1}) present inside network community N_{c_1} and node $(S_{n_{1+n}}, N_{c_2})$ present inside network community N_{c_2}.

$$\begin{aligned} F_{N_{c_{1-2}}} &= (S_{n_1}, N_{c_1}, S_{n_{1+n}}, N_{c_2}) | \exists (N_{c_1} : N_{c_2}) \\ &|(N_{c_1} = (v_1, e_1), N_{c_2} = (v_2, e_2)) | (N_{c_1} = N_{c_2}) \\ &\Rightarrow (S_{n_1}, N_{c_1}) \in v_1 \wedge (S_{n_{1+n}}, N_{c_2}) \in v_2 \\ &\subseteq F_{N_{c_{1-2}}} \in (e_1, (N_{c_1} - N_{c_2})) \end{aligned} \tag{4}$$

Lastly, the third failure recovery scenario shows the multiple link multi-community failure recovery scenario, when the failed links belong to four different nodes present inside two different network communities (i.e node $(S_{n_1}, N_{c_1}, S_{n_{1+n}}, N_{c_1})$ belongs to network community N_{c_1} and nodes $(S_{n_n}, N_{c_2}, S_{n_{+n}}, N_{c_2})$ belongs to network community N_{c_2}. Multiple link failure scenario inside two network community N_{c_1} and N_{c_2} is shown in equation 5.

$$\begin{aligned} F_{N_{c_1}} - F_{N_{c_2}} &= \{(S_{n_1}, N_{c_1}, S_{n_{1+n}}, N_{c_1}), (S_{n_n}, N_{c_2}, \\ S_{n_{+n}}, N_{c_2})\} &| \exists (N_{c_1} : N_{c_1}, N_{c_2} : N_{c_2}) | (N_{c_1} = (v_1, e_1) \\ &, N_{c_2} = (v_2, e_2)) \subseteq F_{N_c}(F_{N_{c_1}} - F_{N_{c_2}}) \\ &\in (e_1 N_{c_1}, e_2 N_{c_2}) \end{aligned} \tag{5}$$

The contribution of the proposed CDRA scheme is explained in equation 4 and equation 5.

The path obtained by applying the general recovery algorithm without community detection approach implementation from the source node to the destination node is presented in equation 6.

$$P_{D_g} = \{P | \forall (S_n, D_n) \in V : P = D_g(P(S_n, D_n))\} \tag{6}$$

Unlike equation 6, equation 7, represents the shortest path formula for the proposed CDRA scheme after implementing the community detection approach.

$$\begin{aligned} P_{D_{N_c}} = \{P | \forall (S_n, N_{c_S}, D_n, N_{c_D}) \in V : \\ P = D_{N_c}(P(S_n, N_{c_S}, D_n, N_{c_D})\} \end{aligned} \tag{7}$$

A set of notations used for the explanation of the proposed CDRA scheme and algorithms are presented in Table II.

TABLE II. List of Notations

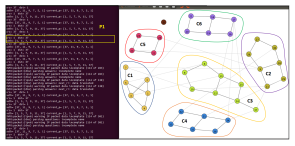
Symbol	Description
P_p / P_s	Primary path / Secondary path
S_n / D_n	Source node / Destination node
N_c	Network community
N_{c_S}	Network community based on source
N_{c_D}	Network community based on destination
F_{N_c}	Failure in network community
D_g	Dijkstra algorithm based on graph
D_c	Dijkstra algorithm based on community
P_{D_g}	Path obtained by Dijkstra algorithm
$P_{D_{N_c}}$	Dijkstra-based on community path

VI. CDRA: An Example Case Study

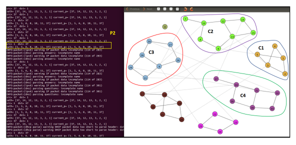
This section, explains the functionality of the proposed CDRA scheme by using the COST266 network topology [26] as an example case study. The Cost266 network topology is made up of 37 nodes and 57 links that connect them. Following the implementation of the community detection algorithms, different colors were utilized to represent the various network communities. Fig 4(a) represents the network topology graph after the implementation of the proposed CDRA scheme before happening link failure scenario. In which a primary path named P1 pass through three different network communities (i.e. N_{c_1}, N_{c_2} and N_{c_3}. The single link intra-community, inter-community and multiple link multi-community failure recovery cases are explained in following three scenarios:

- Single link intra-community failure recovery scenario:

Let suppose the link between nodes 5 and 7 is down. It's worth noting that this is the first failure scenario in which both nodes node 5 and node 7 are members of the same network community N_{c_1} as shown in Fig. 4(a). After happening failure, the network controller calculates a new path "P2" and updates the network topology graph. This time the second path passes through four different network communities (i.e. N_{c_1}, N_{c_2},N_{c_3}, and N_{c_4} as shown in Fig 4(b).



(a) CDRA before Intra Community Link Failure Scenario.

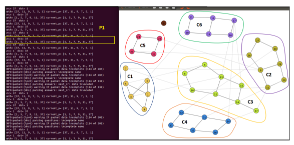


(b) CDRA after Intra Community Failure Scenario.

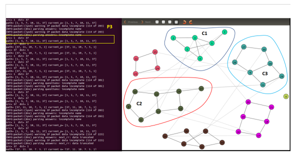
Fig. 4. Single Link Intra Community Failure Recovery Scenario Through CDRA Scheme.

- Inter-community link failure recovery scenario:

In the inter-community link failure recovery scenario, suppose the link between two nodes, node 7 and node 9 fails. It should be noticed that this time both nodes belong to two separate network communities as node 7 belongs to N_{c_1} and node 9 belongs to N_{c_2} as shown in Fig. 5(a). After the occurrence of inter-community link failure scenario the network controller updates the network topology and determine a new path "P3" that passes through three network communities (i.e. N_{c_1}, N_{c_2} and N_{c_3} as shown in Fig 5(b).



(a) CDRA before Inter Community Link Failure Scenario.

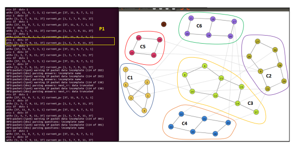


(b) CDRA after Inter Community Link Failure Scenario.

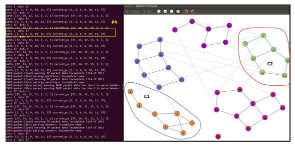
Fig. 5. Inter Community Link Failure Recovery Scenario through CDRA Scheme.

- Multiple link multi-community failure recovery scenario:

Lastly, in the multiple link multi-community failure recovery scenario, imagine a multi-link failure that occurred simultaneously between four separate nodes. This time suppose the first failure occurred between nodes 5 and node 7, which belong to a network community one N_{c_1} , and the second failure occurred between nodes 9 and node 11, both nodes are present inside the network community two N_{c_2} . This time, all four nodes are members of two different network communities N_{c_1} and N_{c_2} as shown in Fig. 6(a). After the occurrence of multiple failures inside two different network community, the network controller creates a new path, "P4", and this time path that passes through two network communities (i.e. N_{c_1}, and N_{c_2} as shown in Fig 6(b).



(a) CDRA before Multi Community Link Failure Scenario.



(b) CDRA after Multi Community Link Failure Scenario.

Fig. 6. Multiple Link Multi-community Failure Recovery Scenario through CDRA Scheme.

This example study indicates that after failure scenarios only a limited number of affected network communities need to be evaluated. The controller just needs to replace and update a few listed nodes that belong to affected communities and have been adopted by new pathways to update the rules. The rest of the nodes, which are dispersed among the various communities, keep their configuration and order. This is how the proposed CDRA scheme makes the failure recovery process more efficient by directly dealing with the failure affected communities and installing a new path, instead of searching through the entire network graph.

VII. Simulation Setup and Result Discussion

To simulate the proposed CDRA scheme the following software tools and programming language is used on the experimental platform: Ubuntu 14.04 LTS is a long-term support version of Ubuntu, Mininet 2.2 developed by Nick McKeown from Stanford University, POX Controller (carp branch), NetworkX, Python 2.7.9 version. The hardware environment includes a PC a 64-bit operating system, x64- based processor DESKTOP-85IQV1U that has an Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz 3.40 GHz as a CPU, 16.0 GB DDR3 1600 of internal user memory. Summary of the simulation setup along with the software tools is presented in Table III.

TABLE III. Summary of the Simulation Setup Along with the Software Tools

Operating system	Ubuntu 14.04 LTS
System Specification	x64-Intel(R)-Core(TM) i7-4770 CPU
simulation Tool	Mininet 2.2
Remote Controller	POX Controller
POX Branch	Carp
OpenFlow Support	OpenFlow v1.0
Programming Language	Python 2.7.9
Network Topology	Atlanta , Cost-266
Bandwidth	10 Mbit/sec
Delay	1 ms
Packet Size (byte)	64

Furthermore, a system work flowchart is presented in Fig. 7, which depicts the simulation setup for the proposed CDRA recovery scheme and the general recovery scheme.

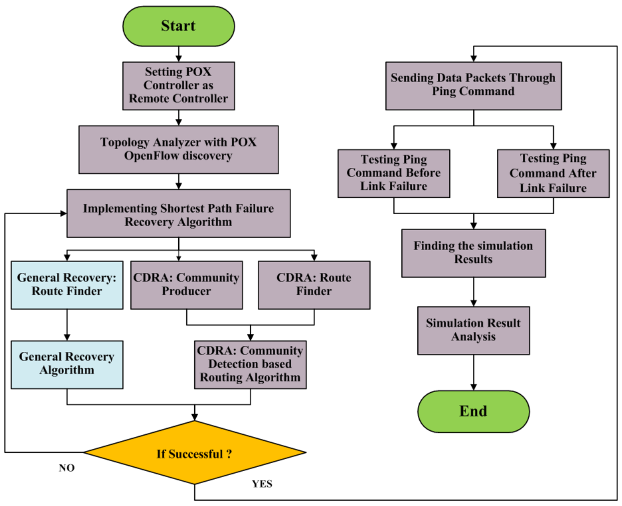

Start
Setting POX Controller as Remote Controller
Topology Analyzer with POX OpenFlow discovery
Implementing Shortest Path Failure Recovery Algorithm
General Recovery: Route Finder
CDRA: Community Producer
CDRA: Route Finder
General Recovery Algorithm
CDRA: Community Detection based Routing Algorithm
If Successful ?
NO
YES
Sending Data Packets Through Ping Command
Testing Ping Command Before Link Failure
Testing Ping Command After Link Failure
Finding the simulation Results
Simulation Result Analysis
End

Fig. 7. System Work Flowchart for the Proposed CDRA Scheme and the General Recovery Algorithm.

A. Performance Evaluation

This paper examines the performance of the general recovery algorithm and the proposed CDRA scheme after the occurrence of link failure scenarios. We chose two topologies from SNDlib [26] to function as experimental topologies (Atlanta and Cost266), with the scale increasing from Atlanta to Cost266. To simulate these topologies, we utilize Mininet, and the Pox controller to monitor and operate the network. Both recovery algorithms are implemented in the POX controller and studied and measured by the average round trip time (RTT), average data packet loss rate and the average end-to-end delay as performance metrics. Furthermore, this study conducted the simulation results multiple times and calculated their average results with a possible 95 percent confidence interval. The performance metrics simulation results study for both recovery algorithms is discussed in the next section.

B. Average Round-Trip Time (RTT)

The RTT (round trip time) refers to the time that an ICMP (internet control message protocol) data packet takes to travel from source to destination, as well as the time it takes for the destination to acknowledge receipt of the packet. The average RTT can be measured by dividing the complete amount of time by the sum of total RTT by the network server and the client. The average RTT measurements for both the general recovery algorithm and the proposed CDRA scheme after the occurrence of single link intra-community failure, inter-community link failure, and multiple link multi-community failure events are shown in Fig. 8, Fig. 9, and Fig. 10, respectively.

Simulation results show that the values of average RTT after the occurrence of the single link intra-community failure, inter-community link failure, and multiple link multi-community failure scenarios are lower for the proposed CDRA scheme. However, the values of average RTT after the occurrence of link failure scenarios are higher for the general recovery algorithm. In a small network topology like Atlanta, the average RTT difference is not as much but when it comes to large network topologies like Cost266 the the performance difference of average RTT after failure scenarios for both algorithms is clearly visible. This is because the proposed CDRA scheme is more optimized and efficient for path-finding after failure scenarios in the large network than the general recovery algorithm. Because the proposed CDRA scheme split the whole network graph into small communities and hunt for the special the failure affected communities. But on the other hand, the general recovery algorithm search throughout the network graph for path-finding after the occurrence of failure scenarios which is time costing.

Furthermore, simulation results also reflects the community detection methods comparison in term of the average RTT after happening link failures for intra-community, inter-community, and multi communities. Simulation results also show that the Louvain and Infomap community detection approaches perform slightly better than the Girvan and Newman community detection approaches. This is because the Girvan and Newman community detection approach is a bit time-consuming and not a preferred approach over a large network. Unlike the Girvan and Newman community detection approach, the Louvain and Infomap community detection approaches are mostly considered to be more optimized approaches for community detection. Research the study supports our results as the Girvan and Newman community detection approach consumes slightly larger time and Louvain community detection approach is more favorable while considering large network graphs [17]–[19].

C. Average Data Packet Loss

Data Packet loss happens when one or more data packets roaming over a computer network do not arrive at their required destination. Network congestion, hardware difficulties, and software faults are all major causes of data packet loss. The ping command is used to send a large number of ICMP data packets from the source node to the destination node, and collect the unsuccessful responses to compute the average data packet loss. Figure 11, Figure 12, and Figure 13 respectively shows the average data packet loss rate readings after the occurrence of intra-community, inter-community, and multi-community link failure scenarios for both the general recovery algorithm and the proposed CDRA scheme.

Simulation results depict that the average value of data packet loss for the general recovery algorithm is higher than the average value of data packet loss achieved by the proposed CDRA recovery scheme. The reason behind this phenomenon is that the general recovery algorithm is not as time-efficient as compared to the proposed CDRA scheme. Because, after happening link failure scenarios, the general recovery algorithm takes longer to search across the whole graph for data packet re-transmission after the event of a link failure. On the other hand, the proposed CDRA scheme, which works only with a small number of failures affected tiny communities in big network graphs and takes less time. Taking a longer time for data packet transmission result in more data packet loss.

In addition, simulation results also reveal that when compared to the Girvan and Newman community techniques, the Infomap and Louvain community methods have a lower or equivalent average data packet loss rate. This is because the Louvain and Infomap community techniques are faster and more efficient than the time-consuming Girvan and Newman method.

D. Average End-to-End Delay

The time it takes for a packet to go from source to destination via a network is referred to as end-to-end delay. The end-to-end delay is determined by dividing the time when data is received by the time it

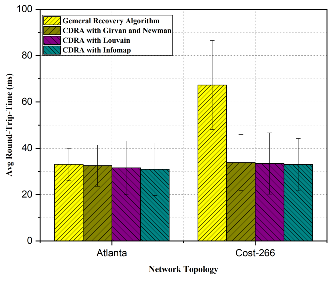


Fig. 8. Average RTT after Occurrence of Single Link Intra-community Link Failure Scenario.

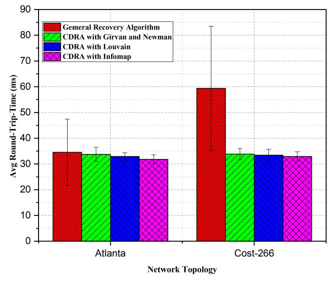


Fig. 9. Average RTT after Occurrence of Inter-community Link Failure Scenario.

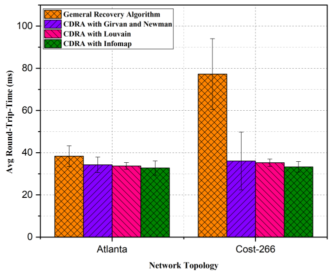


Fig. 10. Average RTT after Occurrence of Multiple Link Multi-community Link Failure Scenario.

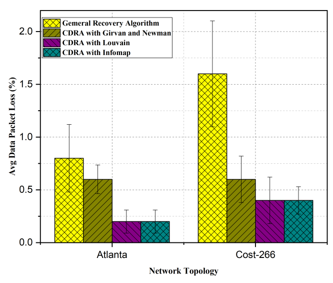


Fig. 11. Average Data Packet Loss after Occurrence of Single Link Intra-community Link Failure Scenario.

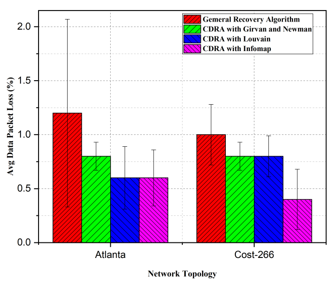


Fig. 12. Average Data Packet Loss after Occurrence of Inter-Community Link Failure Scenario.

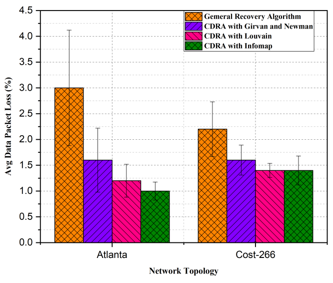


Fig. 13. Average Data Packet Loss after Occurrence of Multiple Link Multi-community Link Failure Scenario.

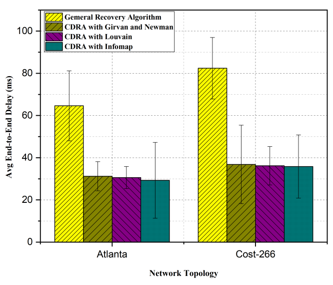


Fig. 14. Average End-to-end Delay after Occurrence of Single Link Intra-community Link Failure Scenario.

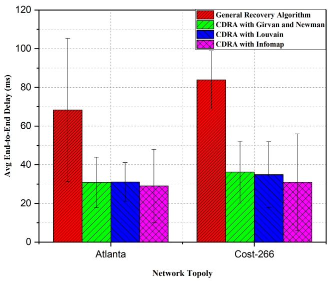


Fig. 15. Average End-to-end Delay after Occurrence of Inter-Community Link Failure Scenario.

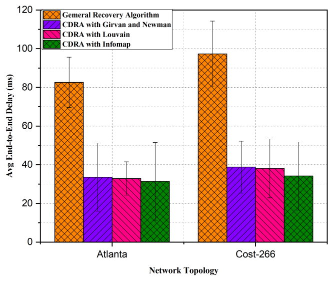


Fig. 16. Average End-to-end Delay after Occurrence of Multiple Link Multi-community Link Failure Scenario.

takes to transmit data by the number of data packets received. This paper measured the average end-to-end delay for both the general recovery algorithm and the proposed CDRA scheme after the occurrence of the link failures scenarios for intra-community, inter-community, and multi communities.

Simulation results of Fig. 14, Fig. 15, and Fig. 16, respectively shows that average end-to-end delay results obtained by the general recovery, algorithm are higher than the average end-to-end delay results obtained by the proposed CDRA scheme. The reason behind these simulations results is the same as we discussed earlier for the average RTT and average data packet loss that the general recovery algorithm takes more time than the proposed CDRA recovery scheme. Because the proposed CDRA recovery scheme split the network graph into small network communities and in case of failure CDRA approach deals only with the failure affected community and the rest of the network graph is not disturbed.

However, on the other hand, the general recovery algorithm considers the entire graph for correction after the occurrence of failure scenarios which make it time cost which result in larger end-to-end delay.

Furthermore, simulation results also shows that due to the time complexity of the Girvan and Newman community detection method, the proposed CDRA scheme shows higher end-to-end delay results as compared to Louvain and Infomap community detection methods which are more effective and time-efficient community detection methods.

VIII. Conclusion and Future Work

Due to the increasing complexity and demand of Internet usage, a new notion of Software Defined Network (SDN) technology has emerged. SDN technology eliminates the limitations of traditional networks and allows IP networks to be programmed. SDN, like traditional networks, is vulnerable to network failures. Link failure is the most prevalent network failure in both traditional networks and SDN. Several studies have been undertaken to improve the speed and efficiency of the link failure recovery procedure. One of these research studies is the use of a community detection mechanism for SDN failure recovery.

However, several specific difficulties, such as multiple link failure and inter-community link failure, were not addressed in these studies.

This work developed a community detection-based routing algorithm (CDRA) method that can handle intra-community, inter-community, and multiple-community connection failure scenarios. Using community detection methods, the suggested CDRA scheme partitioned the network graph into smaller communities. In the event of similar failure scenarios, the proposed CDRA system only deals with the failure-affected communities, and the failure-affected nodes present in that failure-affected community are replaced by rules are replaced the failure affected nodes present in that failure affected community and the rest of the communities will remain on their working phenomena. This makes the proposed CDRA recovery scheme more time-efficient than the general recovery algorithm.

Simulation results show that the proposed CDRA scheme shows better performs than the general recovery algorithm. Furthermore, this study also presents the comparative analysis of different community detection methods such as Girvan and Newman, Louvain, and Infomap community detection methods. Simulation results show that the Louvain and Infomap community detection methods have better performance when they are compared with the Girvan and Newman community detection approach. Because the Girvan and Newman community detection approach is slow and time costing and not recommended for a large network. However, the Louvain and Infomap community detection is more efficient and capable of dealing with large network graphs.

The proposed CDRA approach has broad prospects for further development. The next step of this research study is to imply machine learning techniques in proposed CDRA scheme to provide a fast routing solution for SDN restoration.

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[2] A. Rehman, R. L. Aguiar, and J. P. Barraca, "Fault-tolerance in the scope of software-defined networking (sdn)," *IEEE Access*, vol. 7, pp. 124 474–124 490, 2019.

[3] J. Ali, G.-M. Lee, B.-h. Roh, D. K. Ryu, and G. Park, "Software-defined networking approaches for link failure recovery: A survey," *Sustainability*, vol. 12, no. 10, p. 4255, 2020.

[4] P. C. Fonseca and E. S. Mota, "A survey on fault management in software-defined networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2284–2321, 2017.

[5] B. Isyaku, K. A. Bakar, M. S. Mohd Zahid, E. H. Alkhammash, F. Saeed, and F. A. Ghaleb, "Route path selection optimization scheme based link quality estimation and critical switch awareness for software defined networks," *Applied Sciences*, vol. 11, no. 19, p. 9100, 2021.

[6] A. Malik, B. Aziz, C.-H. Ke, H. Liu, and M. Adda, "Virtual topology partitioning towards an efficient failure recovery of software defined networks," in *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 2. IEEE, 2017, pp. 646–651.

[7] A. Malik, R. de Fréin, and B. Aziz, "Rapid restoration techniques for software-defined networks," *Applied Sciences*, vol. 10, no. 10, p. 3411, 2020.

[8] B. Isyaku, M. S. Mohd Zahid, M. Bte Kamat, K. Abu Bakar, and F. A. Ghaleb, "Software defined networking flow table management of openflow switches performance and security challenges: A survey," *Future Internet*, vol. 12, no. 9, p. 147, 2020.

[9] C. Wang, D. Zhang, L. Zeng, E. Deng, J. Chen, and W. Zhao, "A novel mtj-based non-volatile ternary content-addressable memory for high-speed, low-power, and high-reliable search operation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 4, pp. 1454–1464, 2018.

[10] Y. Wang, S. Feng, H. Guo, X. Qiu, and H. An, "A single-link failure recovery approach based on resource sharing and performance prediction in sdn," *IEEE Access*, vol. 7, pp. 174 750–174 763, 2019.

[11] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling fast failure recovery in openflow networks," in *2011 8th International Workshop on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2011, pp. 164–171.

[12] M. Y. Daha, M. S. M. Zahid, K. Husain, and F. Ousta, "Performance evaluation of software defined networks with single and multiple link failure scenario under floodlight controller," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE, 2021, pp. 959–965.

[13] S. A. Astaneh and S. S. Heydari, "Optimization of sdn flow operations in multi-failure restoration scenarios," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 421–432, 2016.

[14] 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.

[15] V. Muthumanikandan and C. Valliyammai, "Link failure recovery using shortest path fast rerouting technique in sdn," *Wireless Personal Communications*, vol. 97, no. 2, pp. 2475–2495, 2017.

[16] S. Rahiminejad, M. R. Maurya, and S. Subramaniam, "Topological and functional comparison of community detection algorithms in biological networks," *BMC bioinformatics*, vol. 20, no. 1, pp. 1–25, 2019.

[17] F. R. Khawaja, J. Sheng, B. Wang, and Y. Memon, "Uncovering hidden community structure in multi-layer networks," *Applied Sciences*, vol. 11, no. 6, p. 2857, 2021.

[18] S. Emmons, S. Kobourov, M. Gallant, and K. Borner, "Analysis of network clustering algorithms and cluster quality metrics at scale," *PLoS One*, vol. 11, 2016.

[19] Y. L. Youngho Lee, A. S. Jeong Seong, and C. S. Hwang, "A comparison of network clustering algorithms in keyword network analysis: A case study with geography conference presentations," *International Journal of Geospatial and Environmental Research*, vol. 7, no. 3, 2020.

[20] Z. Lu, J. Wahlström, and A. Nehorai, "Community detection in complex networks via clique conductance," *Scientific reports*, vol. 8, no. 1, pp. 1–16, 2018.

[21] V. Rosset, M. A. Paulo, J. G. Cespedes, and M. C. Nascimento, "Enhancing the reliability on data delivery and energy efficiency by combining swarm intelligence and community detection in large-scale wsns," *Expert Systems with Applications*, vol. 78, pp. 89–102, 2017.

[22] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[23] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[24] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[25] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the national academy of sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[26] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[27] K. Mkhitaryan, J. Mothe, and M. Haroutunian, "Detecting communities from networks: comparison of algorithms on real and synthetic networks," *International Journal Information Theories and Applications*, vol. 26, 2019.

[28] R. George, K. Shujaee, M. Kerwat, Z. Felfli, D. Gelenbe, and K. Ukuwu, "A comparative evaluation of community detection algorithms in social networks," *Procedia Computer Science*, vol. 171, pp. 1157–1165, 2020.