Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Network coding-based resilient routing for maintaining data security and availability in Software-Defined Networks

Haoran Ni [a], Zehua Guo [a,*], Changlin Li [a], Songshi Dou [a], Chao Yao [b], Thar Baker [c]

[a] Beijing Institute of Technology, China
[b] Shaanxi Normal University, China
[c] University of Sharjah, United Arab Emirates

ARTICLE INFO

Keywords:
Data security
Data availability
Software-Defined Networking
Routing

ABSTRACT

Software-Defined Networking (SDN) improves network performance by flexible traffic control. Data security and data availability are two main concerns for designing a resilient routing algorithm in SDN. Existing algorithms such as the MPT algorithm consider joint data security and availability, but they cannot make a good trade-off. In this paper, we propose a Network Coding-based Resilient Routing algorithm named NCRR to jointly achieve data security and availability. NCRR is a heuristic algorithm that computes routing decisions in three scenarios based on the number of disjoint paths. Specifically, the scenario of three disjoint paths is enough to ensure joint security and availability for a flow when three or more disjoint paths can be used for forwarding this flow. However, due to topological diversity, we cannot always find three disjoint paths for each flow. Thus, the scenarios of two disjoint paths and one path are used to ensure joint security and availability if only two disjoint paths and one path can be used. To evaluate the performance of NCRR, simulations have been conducted using two real-world network topologies. Simulation results show that NCRR improves the joint data security and availability performance by approximately 6.23% on AttMpls topology and 21.34% on Cernet topology, compared with existing MPT.

1. Introduction

Software-Defined Networking (SDN) has substantially changed traditional network design, management, and operations (Rischke and Salah, 2021). It facilitates programmable networks in a bid to provide flexible network traffic control and dynamic configuration. SDN allows runtime adaptable behavior of the network using well-defined tools, APIs, and interfaces. It also helps minimize the overall cost to network service providers and enhance the network quality for their subscribers and users.

SDN decouples the control plane from the data plane via an open API (i.e., OpenFlow protocol McKeown et al., 2008). Put simply, OpenFlow allows remote control of packet forwarding engines to gain unprecedented programmability and control over network devices by deploying high-level policies in lieu of conventional low-level device configurations. One major benefit of SDN comes from path programmability, which enables network service providers to design customized routing solutions to adaptively change forwarding paths for individual flows based on dynamic network states.

Data resiliency is usually considered into designing the dynamic routing solution by distributing segments to multiple paths (Mohan et al., 2018). When routed data traverses through a few compromised nodes, many sensitive data traversing the compromised nodes may be dropped or intercepted (Juan et al., 2021). Data resiliency ensures that data can be fully recovered anytime even if it experiences accidental loss or is maliciously stolen during transmission. One potential solution is to employ Network Coding (NC) to add data redundancy (Juan et al., 2021). At the source node, data can be fragmented and encoded with an encoding algorithm, such as Maximum Distance Separable (MDS) block code, which preserves data patterns. K data blocks can be redundant to N data blocks where N is larger than K with MDS coding, while we only need K data blocks to reconstruct the entire file (Sathiamoorthy et al., 2013). The encoded data are distributedly forwarded to the destination node, which decodes the received data to fully recover the data.

However, existing solutions suffer from three main issues: (i) low data availability, which cannot guarantee that transmitted data can always reach the destination; (ii) high overhead, which increases the network burden; and (iii) the trade-off between data security and availability (Scott-Hayward et al., 2013). To increase data availability, we distribute segments on several paths. Hence, when one node is

* Corresponding author.
E-mail address: guolizihao@hotmail.com (Z. Guo).

https://doi.org/10.1016/j.jnca.2022.103372
Received 10 June 2021; Received in revised form 26 January 2022; Accepted 21 March 2022

compromised, we can still receive enough segments from other secure paths to reconstruct the data. However, using more segments and paths could increase the risk of data leakage because the attacker has more opportunities to fetch the data, which leads to decreasing data security. Multipath-Tree Based Heuristics (MPT) (Mohan et al., 2018) is a state-of-the-art solution that allocates segments by establishing a path tree and finding the most reliable multipath solution. However, MPT focuses more on security than availability and introduces higher average overhead.

In this paper, we propose Network Coding-based Resilient Routing (NCRR) to achieve a trade-off between data routing security and availability in SDNs. On the premise of ensuring security, we endeavor to maximize availability and make overhead relatively low. In some scenarios, NCRR guarantees 100% security and availability at the same time. While in other scenarios where a trade-off between data security and availability cannot be guaranteed, there is also one path algorithm and screening algorithm of the path-set, which can improve the overall performance. It is critical to distribute data segments to available paths for routing security. In our algorithm, we divide the problem into three scenarios, which cover all cases. Specifically, the scenario of three disjoint paths is enough to ensure joint security and availability for a flow when three or more disjoint paths can be used for forwarding this flow. However, due to topological diversity, we cannot always find three disjoint paths for each flow. Thus, the scenarios of two disjoint paths and one path are used to ensure joint security and availability if only two disjoint paths and one path can be used.

The novelty of our algorithm is to achieve a better trade-off between data security and availability with low overhead. We aim to forward data reliably, securely, and efficiently. We use three scenarios for NCRR. The scenario of three disjoint paths is enough to ensure joint security and availability for a flow when three or more disjoint paths can be used for forwarding this flow. However, due to topological diversity, we cannot always find three disjoint paths for each flow. Thus, the scenarios of two disjoint paths and one path are used to ensure joint security and availability if only two disjoint paths and one path can be used.

The contributions of this paper can be summarized as follows:

- We formulate the Joint Security and Availability Problem (JSAP), which is a Mixed-Integer Programming (MIP), for realizing data security and data availability together.
- We propose a novel routing algorithm named NCRR to efficiently solve the problem. NCRR works for three specific scenarios based on the number of disjoint paths. For each scenario, NCRR selects forwarding paths and decides the number of segments on the selected paths for each flow to achieve jointly data security and availability in SDN.
- We evaluate the performance of NCRR under two real-world network topologies. Simulation results show that NCRR improves data security and availability performance by approximately 6.23% on AttMpls topology and 21.34% on Cernet topology (Knight et al., 2011), compared with the existing MPT algorithm.

The rest of this paper is arranged as follows. The research and development background and the limitations of the existing solutions are analyzed in detail in Section 2. In Section 3, we introduce the theoretical design of NCRR. Section 4 gives the concrete implementation of the algorithm, and Section 5 explains the results and analysis of the simulation experiment in detail. Finally, we review the relevant work in Section 6 and draw a conclusion in Section 7.

2. Background and motivation

2.1. Network coding-based secure routing

Network coding-based resilient routing can have multiple types of implementations. Fig. 1 illustrates an implementation structure using Network Function Virtualisation (NFV) (Mohan et al., 2018). The source server sends K original segments to the source switch (SW), which follows rule 1 to forward the K segments to the source NFV. The source NFV uses its network coding module to encode the original K segments to N Network Coding (NC) segments and then sends a request about the routing decisions on the N NC segments to the SDN controller. In the SDN controller, the routing selection module assigns a forwarding path for a selected number of segments. Upon receiving the NC segments from the source NFV and the routing decisions from the SDN controller, the source SW adds rule 2 to store the routing decisions and forwards the N NC segments on the selected M paths. Upon receiving the NC segments, the destination SW follows rule 1 to forward the segments to the destination NFV to decode the NC segments to the original segments and follows rule 2 to finally forward the decoded segments to the destination server.

In this example, we mainly have four segments, and the source NFV changes the four original segments into six NC segments. The NC segments are distributed on three paths: three segments on path p1: $SW1 \rightarrow SW2 \rightarrow SW3$, two segments on path p2: $SW4 \rightarrow SW5 \rightarrow SW6$, and one segment on path p20: $SW40 \rightarrow SW5 \rightarrow SW60$. When SW4 is compromised, the attacker receives two NC segments and the destination SW/NFV receives four NC segments. Since decoding the NC segments requires at least four NC segments, the destination NFV can decode the NC segments to the original segments but the attacker cannot. Routing prevents the attacker from obtaining the information.

2.2. Requirements for secure routing

We use an example to illustrate the motivation. If one node is compromised, we call the path using the compromised node a compromised path , and other paths without the compromised node the uncompromised paths.

2.2.1. Data security

The security requirement ensures an attacker cannot decode the flow. Since the requirement of decoding NC segments is to have at least K NC segments, in this example, the requirement should prevent each node on the forwarding paths from having more than four NC segments.

2.2.2. Data availability

We need another requirement to ensure that the destination server has the original segments of a flow. That is, let the uncompromised paths forward at least K NC segments to the destination SW. We call it the availability requirement. For this example, the availability requirement is to let the uncompromised paths forward at least four NC segments.

2.2.3. Routing overhead

This index reflects the overhead cost of the routing algorithm. It is the ratio of the number of NC segments after redundancy to that before redundancy. In this example, the overhead ratio is $\frac{6}{4}$, namely, 1.5.

2.3. Limitation of existing solutions

2.3.1. No data availability guarantee

Multipath-Tree Based Heuristics (MPT) (Mohan et al., 2018) mainly consider the security requirement.

The example in Fig. 2 is the same example in Fig. 1. That is $K = 4$, $N = 6$, and $M = 3$. In the example, the source server sends four segments of one flow to the destination server. Due to the limited space, we do not show the source server, source NFV, destination NFV, or destination server. The function of the two servers and two NFVs are same as the ones in Fig. 1.

Fig. 2(a) shows the result of using MPT when the attacker compromises one node. In Fig. 2(a), path p1: $SW1 \rightarrow SW2 \rightarrow SW3$, path p2: $SW4 \rightarrow SW5 \rightarrow SW6$ and path p20: $SW40 \rightarrow SW5 \rightarrow SW60$ for-

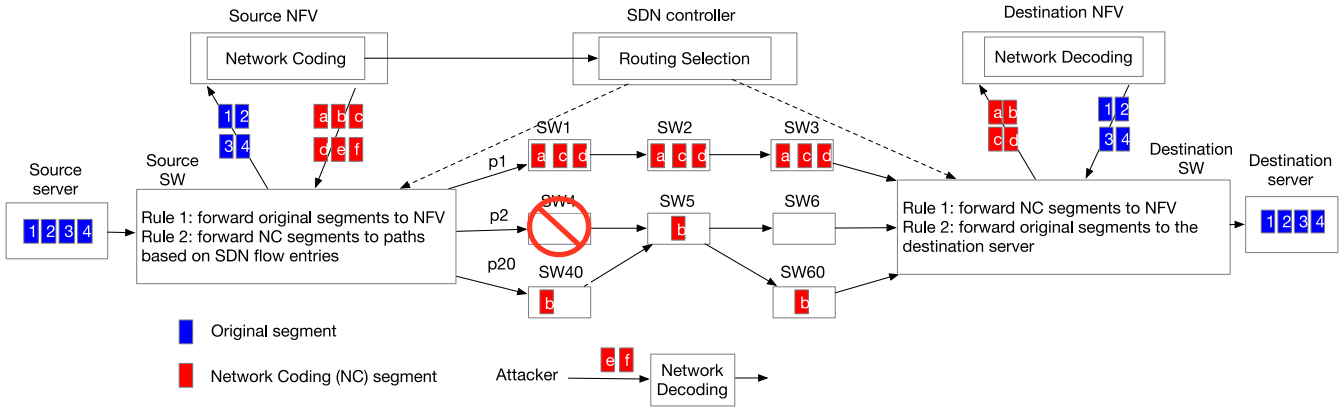
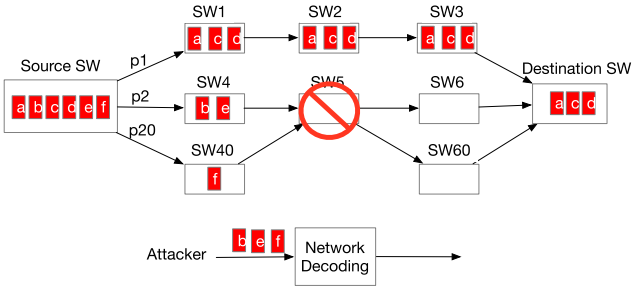
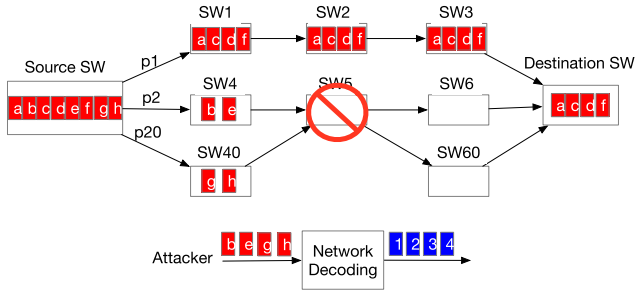


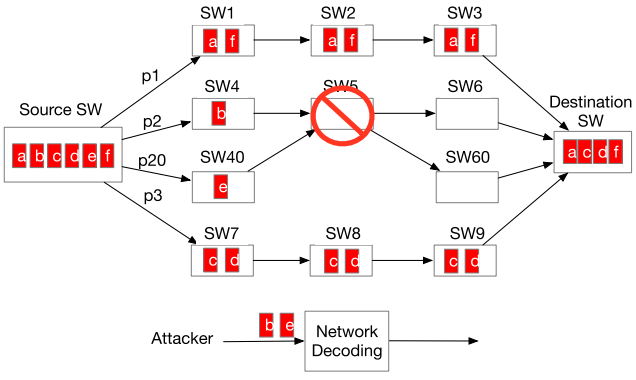
Fig. 1. Structure of Network Coding-based Resilient Routing using NFVs (Mohan et al., 2018). The number of original segments is $K = 4$, and the number of NC segments is $N = 6$. The source NFV uses its network encoding module to transform the four original segments to six NC segments. The six NC segments are distributed into three paths. NC segments 'a', 'c', and 'd' are transmitted through SW1. NC segments 'e' and 'f' are transmitted through SW4, and 'b' is transmitted through SW40. The requirement of decoding the NC segments is to have at least K NC segments. When SW4 is compromised, the destination NFV receives four NC segments and can decode the NC segments to the original segments, while the attacker receives only two NC segments and cannot decode the NC segments.



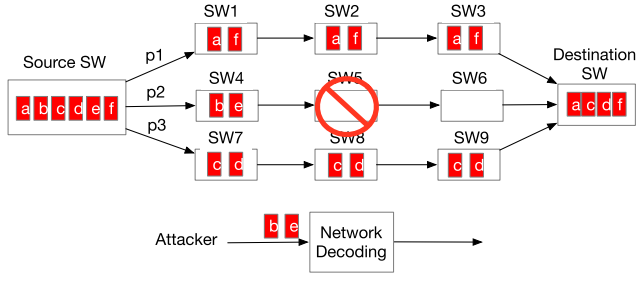
(a) The result of only considering the security requirement. NC segments 'a', 'c', and 'd' are transmitted through SW1. NC segments 'b' and 'e' are transmitted through SW4, and 'f' is transmitted through SW40. When SW6 is compromised, the destination server and the attacker both receive three NC segments and cannot decode the NC segments to the original segments.



(b) The result of only considering the availability requirement. NC segments 'a', 'c', 'd' and 'f' are transmitted through SW1. NC segments 'b' and 'e' are transmitted through SW4, and 'g' and 'h' are transmitted through SW40. When SW5 is compromised, the destination server and the attacker both receive four NC segments and can decode the NC segments to the original segments.



(c) The result of jointly considering the security requirement and availability requirement. NC segments 'a' and 'f' are transmitted through SW1. NC segment 'b' is transmitted through SW4. NC segment 'e' is transmitted through SW40, and 'c' and 'f' are transmitted through SW7. When SW5 is compromised, the destination server receives four NC segments and can decode the NC segments to the original segments, while the attackers receives only three NC segments and cannot decode the NC segments.



(d) The result of jointly considering the security requirement and availability requirement and involving the minimum number of paths. NC segments 'a' and 'f' are transmitted through SW1. NC segments 'b' and 'e' are transmitted through SW4, and NC segments 'c' and 'd' are transmitted through SW7

Fig. 2. A motivation example when the attacker compromises one node.

ward three segments, two segments and one segment, respectively. Path p2 and path p20 are overlapped at SW5. When SW5 is compromised, paths p2 and p20 become compromised paths, one segment from path p2 and two segments from path p20 can be acquired by the attacker. Since the minimum number of segments for decoding the flow is four, the attacker cannot decode the flow. However, the destination server cannot decode the flow because it also only obtains three segments. A similar situation occurs when SW2 is compromised. In other words, only considering the security requirement cannot guarantee that the destination NFV receives enough NC segments to decode the flow.

Fig. 2(b) shows an example of the MPT. In this example, MPT ensures availability when any node in the three paths is compromised.

However, in the figure, the attacker can also acquire four segments to decode the flow.

2.3.2. High routing overhead

The routing resiliency increases with NC-based routing at the cost of using more forwarding paths. Fig. 2(c) uses four paths: p1, p2, p20, and p3. Using multiple paths to forward a flow could affect the forwarding of other flows. To reduce the undesirable impact, we should achieve the two requirements with fewer paths. Fig. 2(d) shows an example that only uses three paths: p1, p2, and p3. We can see that the number of paths for forwarding NC segments and the distribution of NC segments on the paths affect both the requirements and the routing overhead. Thus, our problem should consider both factors to achieve the two requirements with the minimum routing overhead.

2.4. Design consideration

A feasible solution should jointly achieve security requirement and availability requirement. Fig. 2(c) shows an example solution that jointly achieves the two requirements. In the figure, we use four paths and reasonably distribute six NC segments into the four paths. With the solution, the attacker can acquire at most two NC segments when SW5 or any one node in the four paths is compromised, while the destination can always obtain at least four NC segments.

3. Problem formulation

In this section, we first introduce the constraints and the objective function and then formulate our optimization problem.

3.1. System description

The network is graph $G = (V, E)$. A flow has K segments, and the segments are encoded with the network coding to $N(N > K)$ NC segments. All paths of the flow generate the path-set P^{all}, and the N NC segments are distributed to a path-set $P(P \subset P^{all})$ with a selected number of paths from P^{all}. If path p with length l_p in terms of hops is selected, $x_p = 1$ and y_p ($y_p \leq K$) segments are forwarded on path p; otherwise, $x_p = 0$ and $y_p = 0$. The relationship between of y_p and x_p is

$$y_p * x_p = y_p. \tag{1}$$

A path consists of some nodes and links. If node v ($v \in V$) belongs to path p, we have $\phi_{v,p} = 1$; otherwise, $\phi_{v,p} = 0$. If link e ($e \in E$) belongs to path p, $\alpha_{e,p} = 1$; otherwise $\alpha_{e,p} = 0$. The set of paths in P relates to node v is P^v. Compromising node v can acquire all the segments of paths traversing v. We call the paths related to a compromised node v the compromised path-set of v.

3.2. Constraints

3.2.1. Transmission constraint

All N NC segments of a flow should be forwarded on all selected paths of the flow:

$$\sum_{p \in P^{all}} y_p = N. \tag{2}$$

3.2.2. Security constraint

An attacker can decode a flow if he can obtain K NC segments. We assume an attacker knows the network coding is used for transmitting packets, and any node can be compromised by the attacker. To prevent the attacker from decoding the flow, we need to ensure that the total number of NC segments of a flow on any node should be less than K. That is:

$$\sum_{p \in P^{all}} y_p * \phi_{v,p} < K, \forall v \in V. \tag{3}$$

3.2.3. Availability constraint

To recover a flow, at least K NC segments of the flow should arrive at the destination SW through uncompromised paths when any one node is compromised. When no node is compromised, the destination SW receives N segments. If node v is compromised, the number of lost segments on the compromised node is $\sum_{p \in P^v} y_p * \phi_{v,p}$. Thus, when any one node $v (v \in V)$ is compromised, the number of NC segments that the destination server receives should be more than K. That is:

$$N - \sum_{p \in P^{all}} y_p * \phi_{v,p} \geq K, \forall v \in V. \tag{4}$$

3.2.4. Joint security and availability constraint

If we only have two paths, the availability constraint requires each node of the two paths to have K NC segments, which contradicts the security constraint. Thus, we need at least three paths. That is

$$\sum_{p \in P^{all}} x_p \geq 3. \tag{5}$$

3.2.5. Bandwidth constraint

The bandwidth constraint ensures the availability of bandwidth on the links of a selected path p. For any link e belonging to path p, it must have sufficient residual bandwidth to accommodate the assigned NC segments of the flow. That is:

$$y_p * \alpha_{e,p} < B_l^{rest}, \forall e \in E, p \in P^{all}. \tag{6}$$

where B_l^{rest} is the residual bandwidth on link e and can be updated in the SDN controller in real time.

3.2.6. Delay constraint

For path p for routing a flow, the total delay in terms of hops should be no longer than a given threshold L. That is:

$$\sum_{e \in E} \alpha_{e,p} * x_p \leq L, \forall p \in P^{all}. \tag{7}$$

3.3. Objective function

We define the total cost of using NC-based routing equal to the number of total NC segments on the selected paths. That is:

$$Total\_cost = \sum_{p \in P^{all}} (l_p * x_p) \tag{8}$$

3.4. Problem formulation

We formulate the joint security and availability problem as follows:

$$min \sum_{p \in P^{all}} (l_p * x_p) \tag{9}$$

subject to

$$(2), (3), (4), (5), (6), (7),$$
$$x_p \in [0, 1], y_p \in N^+, p \in P^{all}, v \in V, e \in E.$$

In the above problem, x_p are binary variables, y_p are integer design variables, $\phi_{v,p}$ and $\alpha_{e,p}$ are given binary constants, and l_p, B_l^{rest}, L, K, and N are given integer constants.

4. Proposed solution

4.1. Problem analysis

As mentioned above, the problem that this paper discusses is deemed a Mixed-Integer Programming (MIP) -based problem with a large solution space. The security requirement in Eq. (3) tests all paths of a flow, and the solution space of the availability requirement is proportional to the total number of paths of a flow and the number

Algorithm 1 ThreeDisjointPaths(T)

Input:
T: The collection of disjoint path-sets, $T = \{Pi | Pi$ is a set of paths related to a disjoint path $p_i\}$

```
path requirements: (1) bandwidth constraint, (2) delay constraint
path-set requirements: each path-set has one path
for P ∈ T do
    FindFeasiblePathSet(P, path requirements, path-set requirements)
    if the number of feasible path-sets equals three then
        P = {paths in all feasible path-sets}
        break;
    end if
end for
for p ∈ P do
    for e ∈ p do
        the load of e + ⌊K/2⌋;
    end for
end for
```

Algorithm 2 FindFeasiblePathSet(P, path requirements, path-set requirements)

Input:
P: a set of paths for a flow
path requirements,
path-set requirements
Output:
P^{new}: a new path-set

```
generate a new path-set P^new = ∞
for p ∈ P do
    if p satisfies path requirements then
        P^new ← P^new ∪ p
    end if
    if path-set satisfies path-set requirements then
        return P^new
    end if
end for
return ∞
```

of nodes in the network as noted in Eq. (4). To efficiently solve the problem, we first analyze the essence of the problem. Essentially, the complexity of the problem comes from the availability requirement. The security requirement only tests individual nodes and is easy to satisfy while the test result of the availability requirement depends on three factors: (1) the placement of the tested node (2) the paths traversing the tested node, and (3) the interaction between selecting paths traversing the tested node and the selection of other paths. To make a decision satisfying the availability requirement, we first decide which node we should select. If selecting it, we need to further determine how many and which paths should be selected for this node since multiple paths can use the node. In addition, the selection of one path on the node could affect the selection of other paths. This is because a selected path could overlap other paths via other nodes, and thus it prevents selecting multiple other paths. Considering the large solution space, the complicated decision process increases the problem's complexity. We should find a way to simplify the complexity of the availability requirement.

The result of the problem is to select a set of paths for a flow, and only one node is compromised in the problem. Assume the path-set of the flow is selected. For a path in the path-set, if compromising any one node in the path only affects the path itself, we only need to achieve the availability requirement on the remaining paths in the path-set. We call it the Path Selection Constraint. Following the constraint, we can simplify the test of the availability requirement by selecting disjoint paths. In other words, we can achieve the availability requirement by finding disjoint paths to generate the set of paths for a flow.

Following the constraint in Eq. (5), to jointly achieve the availability requirement and security requirement, we need to find three disjoint paths. However, the solution does not always work since the network topologies are different, and in some cases, we can only find two disjoint paths for a flow or even worse only one path for a flow. For the scenarios of two disjoint paths or one path, the path selection constraint is too strong. To solve our problem under the two scenarios, we can relax the constraint to the Weak Path Selection Constraint: compromising any one node affects only two paths in the set of selected paths. In other words, we should ensure that in the path-set, any two paths overlap at most at one node, and need to achieve the data availability requirement on the paths in the path-set except two overlapping paths.

4.2. Heuristic algorithm

The network topology is given and does not change very frequently. During the network initialization, we can generate all disjoint path-sets for each pair of source node s and destination node d. We first find all disjoint paths for (s, d), and then generate a path-set for each disjoint path to store all paths related only to the disjoint path. In each disjoint path-set, the first path is the disjoint path, and the other paths are sorted in ascending order of the path length. The collection of disjoint path-sets T is stored in the SDN controller.

Based on the above analysis, we can categorize the solution of the problem under three scenarios: (1) three disjoint paths, (2) two disjoint paths, and (3) one path. The solution of each scenario is shown below:

4.2.1. Three disjoint paths

For this scenario, we need $|T| \geq 3$. In the network, each node has an equal probability of being compromised. Thus, to equally treat each node and achieve the joint two requirements, the simple solution is to evenly distribute the NC segments into the three disjoint paths. Algorithm 1 shows the algorithm's detail.

4.2.2. Two disjoint paths

For this scenario, we need $|T| \geq 2$. In this scenario, we only find two disjoint paths for a flow: $p1$ and $p2$, and the set of paths overlapping with $p1$ and $p2$ with only one disjoint node are $P1$ and $P2$, respectively. As explained before, we need at least three paths. In our solution, two paths in a path-set overlap at one node, and in terms of the data availability, the function of using two overlapping paths in a path-set is the same as one path in the same path-set. Thus, among the two path-sets, one should have at least three paths, and the other should have at least one path. To minimize the number of forwarding paths for a flow, we have $|P1| = 3, |P2| = 1$.

Since each node has an equal probability of being compromised, the simple and efficient method to dispatch NC segments is to let each path in the same path-set have an equal number of segments. We allocate n_1 segments to each path in path-set $P1$ and n_2 segments to each path in path-set $P2$. The transmission constraint can be changed as follows:

$$n_1 * |P1| + n_2 * |P2| \geq N \tag{10}$$

We can change the security constraint as follows:

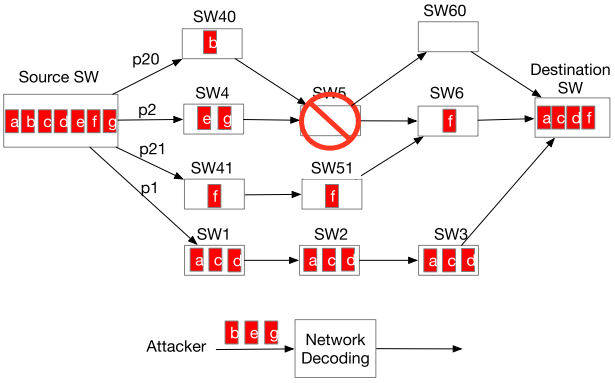
$$n_1 * 2 < K, n_2 < K \tag{11}$$

To guarantee the data availability requirement, we have

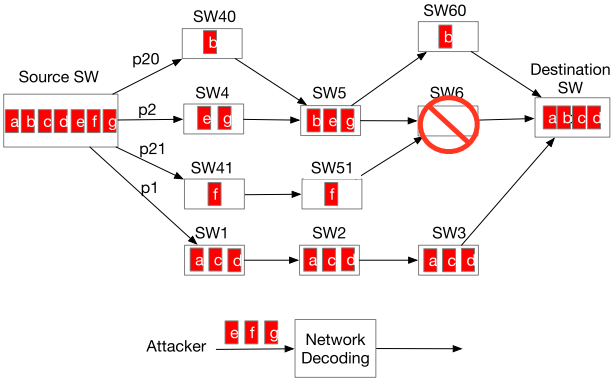
$$n_1 * |P1| \geq K, n_1 * (|P1| - 2) + n_2 \geq K \tag{12}$$

Bring $|P1| = 3$ into the above equations, we have:

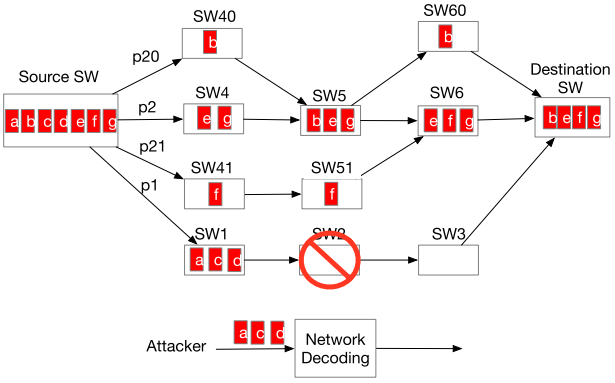
$$n_1 \in [K/3, K/2), n_2 \in [2K/3, K) \tag{13}$$



(a) The result when SW5 is compromised.



(b) The result when SW6 is compromised.



(c) The result SW2 is compromised.

Fig. 3. An example of distributing NC segments to disjoint paths and overlapped paths. Path p2 is overlapped with path p20 at SW5 and p21 at SW6; path p1 is disjoint with the three paths. NC segment 'b' is transmitted through SW40. NC segments 'e' and 'g' are transmitted through SW4. NC segment 'f' is transmitted through SW41, and 'a', 'c' and 'd' are transmitted through SW7. When the attacker compromises any one node, the destination server receives four NC segments and can decode the NC segments to the original segments, while the attacker receives only three NC segments and cannot decode the NC segments.

If n_1 or n_2 has a feasible value, we can use it directly. If not, equal overhead allocation cannot jointly achieve the two requirements. Then we need to test the overhead in $[\lceil K/2 \rceil, \lfloor K/3 \rfloor]$ on paths in the path-set.

Algorithm 3 shows the algorithm's detail. Fig. 3 shows an example of the scenario of two disjoint paths. In the example, we can only have two disjoint paths for a flow: $p2 : SW1 \rightarrow SW2 \rightarrow SW3$, and path $p1 : SW4 \rightarrow SW5 \rightarrow SW6$. Path $p10 : SW40 \rightarrow SW5 \rightarrow SW60$ shares SW5 with $p1$, and path $p11 : SW41 \rightarrow SW51 \rightarrow SW6$ shares SW6 with $p1$. That is, $|P1| = 3$, $|P2| = 1$ and $K = 4$. Using the above result, $n_2 \in [8/3, 4)$, and the feasible value for n_2 is 3. However, $n_1 \in [4/3, 2)$, and n_1 does not have a feasible value. Thus, we need to test overhead $[\lfloor 4/3 \rfloor, \lceil 2 \rceil] = [1, 2]$ for paths $p1$, $p10$, $p11$. We can have a feasible solution for $p1$ with 2 segments, $p10$ with one segment, and $p11$ with one segment. SW5, SW6 and any node in path p1 have the most segments among the paths. In Fig. 3(a), SW5 is compromised, the destination can receive four NC segments, but the attacker only receives three segments. The similar results can be found in Fig. 3(b) when SW6 is compromised and Fig. 3(c) when SW2 is compromised.

4.2.3. One path

If we can find only one path $p1$, we can also achieve the two requirements with the following three equations:

$$n_1 * |P1| \geq N \tag{14}$$

$$n_1 * 2 < K \tag{15}$$

$$n_1 * (|P1| - 2) \geq K \tag{16}$$

Table 1
Constraints on the three scenarios.

Scenario	$\|P1\|$	$\|P2\|$	$\|P3\|$	Length of the first path in $P1$	n_1	n_2	n_3
Three disjoint paths	1	1	1	–	$\lfloor K/2 \rfloor$	$\lfloor K/2 \rfloor$	$\lfloor K/2 \rfloor$
Two disjoint paths	3	1	0	3	[K/3, K/2)	[2K/3, K)	0
One path-like	5 or n	0	0	5 or –	[K/3, K/2) or $\lceil \frac{K}{\|P\|-d} \rceil$	0	0

Algorithm 3 TwoDisjointPaths(T)

Input:
T: The collection of path-sets, $T = \{Pi | Pi$ is a set of paths with a disjoint path $p_i\}$

path requirements of $P1$: (1) bandwidth constraint, (2) delay constraint, (3) the length of the first path equals three, (4) relaxed path selection constraint
path-set requirements of $P1$: the path-set has three paths
path requirements of $P2$: (1) bandwidth constraint, (2) delay constraint
path-set requirements of $P2$: the path-set has one path
for $P \in T$ **do**
    $P1$ = FindFeasiblePathSet(P, path requirements of $P1$, path-set requirements of $P1$)
    if $P1$ exists **then**
        $P\_used = P$;
        break;
    end if
end for
for $P \in T - P\_used$ **do**
    $P2$ = FindFeasiblePathSet(P, path requirements of $P2$, path-set requirements of $P2$)
    if $P2$ exists **then**
        break;
    end if
end for
P1Load,P2Load = CalculateLoad($P1$,$P2$)
for $p \in P1(or P2)$ **do**
    for $e \in p$ **do**
        the load of e + load of p from P1Load (or P2Load);
    end for
end for

Eqs. (14), (15) and (16) are transmission constraints, security constraint and availability constraints, respectively.

From the above equations, we can have:

$$K/2 > n_1 \geq K/(|P1| - 2) \Longrightarrow |P1| - 2 > 2 \Longrightarrow |P1| > 4 \tag{17}$$

If $|P1| = 5$, we can have

$$n_1 \in [K/3, K/2) \tag{18}$$

However, if we cannot find the solution here, we can try to realize the two requirements in another way. For this scenario, we also dispatch NC segments evenly over the paths. Here we firstly define two variables. We denote Defense ability as the number of paths whose packets can be lost at the same time but we can still recover the data and denote Availability ability as the number of paths whose packets can recover the flow when their packets add up. They are denoted by d and a respectively. We also need a set of paths with some degree of dispersion, represented as P. Dangerous nodes refer to those nodes that will affect the security and availability of specific flows after being attacked.

To satisfy the security constraint:

$$a > d \tag{19}$$

To satisfy the availability constraint:

$$|P| - d \geq a \tag{20}$$

Algorithm 4 OneDisjointPath(T, P)

Input:
T: The collection of path-sets, $T = \{Pi | Pi$ is a set of paths with a disjoint path $p_i\}$
P: The collection of dispersed paths, $P = \{p | p$ is a set of paths with some degree of dispersion$\}$

path requirements: (1) bandwidth constraint, (2) delay constraint, (3) the length of the first path equals five, (4) relaxed path selection constraint
path-set requirements: the path-set has five paths
for $P \in T$ **do**
    $P1$ = FindPathSet(P, path requirements, path-set requirements)
    if $P1$ exists **then**
        break;
    end if
end for
if not P1 **then**
    c = collections.Counter()
    for $p \in P$ **do**
        c.update(p)
    end for
    d = Max(c)
    if $d \geq \frac{|P|}{2}$ **then**
        $d = \frac{|P|}{2}$
    end if
    PLoad = CalculateLoad(P, d)
else
    P1Load = CalculateLoad(P1)
end if
for $p \in P1$ **do**
    for $e \in p$ **do**
        the load of e + load of p from P1Load;
    end for
end for

To achieve a better defense effect, d is usually the maximum indegree of nodes d_{max}, but it still needs to satisfy the two constraints above:

$$d = Min(d_{max}, \lfloor \frac{|P|}{2} \rfloor) \tag{21}$$

If the indegree of a node is less than the defense ability, the node is safe, if it is greater than the defense ability, it is dangerous, and if it is equal to the defense ability, it will depend on the parity of the number of paths. Even if a node has indegree d in the path set, we can still receive packets transmitted on $|P|-d$ paths when d is no more than $\frac{|P|}{2}$, and we try our best to defend most nodes on the premise of ensuring the availability constraint when d is more than $\frac{|P|}{2}$. So, numbers of packets dispatched over every path:

$$n = \lceil \frac{K}{|P| - d} \rceil \tag{22}$$

Algorithm 4 shows the algorithm's detail. Table 1 summarizes the requirement on the path length, the number of paths of path-sets, and the overhead of each path of the four scenarios.

The *one disjoint path* algorithm is a good solution, but to achieve better results, we further screened the path set P. In the implementation of the *one disjoint path* algorithm, the indegrees of nodes in the path set

Table 2
Path set before optimization.

Paths	(node index, number of occurrences)
[9, 10, 11, 12] [9, 10, 5, 0, 13, 12] [9, 8, 7, 6, 5, 0, 13, 12] [9, 8, 7, 6, 5, 10, 11, 12]	(0, 2), (6, 2), (7, 2), (8, 2),(11, 2), (13, 2), (5, 3),(10, 3), (9, 4), (12, 4)

Table 3
Path set after optimization.

Paths	(node index, number of occurrences)
[9, 10, 11, 12] [9, 10, 5, 0, 13, 12] [9, 8, 7, 6, 5, 10, 11, 12]	(0, 1), (6, 1), (7, 1), (8, 1), (13, 1), (5, 2), (11, 2), (9, 3), (10, 3), (12, 3)

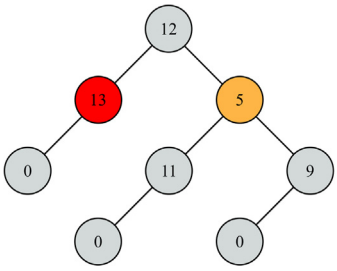


Fig. 4. Path tree of MPT. The red node 13 is dangerous nodes in both NCRR and MPT algorithms, and the orange one 5 is the node with different performance under two algorithms.

determine whether the node affects the security and recoverability of a specific flow. Abstractly speaking, some nodes will appear frequently because of the existence of some paths in the path set P. If we remove some of these paths with nodes whose indegree is near the critical value, namely, the half size of the pathset, it is possible to save more dangerous nodes and improve the data security and availability.

For example, the statistics of path sets and node indegrees are shown in the Tables 2 and 3. Table 2 shows the path set before optimization of a specific flow and the statistical results of nodes. Except for the source and destination nodes, the indegrees of the remaining six nodes are greater than or equal to half of the size of the path set. That is, every node in the middle will affect the security and recoverability of the flow. Table 3 shows the path set after optimization. We removed the third path, and the result was that, except for the source and destination nodes, only three of the remaining six nodes affected the security and recoverability of the flow. The effect of this optimization is obvious.

We preset a few concepts to facilitate the elaboration. Dangerous nodes refer to those nodes that will affect the security and availability of specific flows after being attacked. The process of optimizing the path set is also the process of repairing some dangerous nodes. The order of the node represents the number of paths containing the node in the path set, which is also equal to the indegree of the node in the path set. Since we do not distinguish which nodes are more important, repairing dangerous nodes can be understood as repairing nodes of a certain order.

With the above concept, this Path Set Optimization Process (PSOP) can be composed of three parts, namely, determining the order of repair, determining the number of paths to be abandoned, and determining the paths to be abandoned.

Whether a node is dangerous depends on the relationship between its order and half of the number of paths. Therefore, our repair starts with the order of half the number of paths. We find the relationship between the order of repair and the number of paths to be discarded, as shown in Eq. (23).

$$t - n - r_{t-n} < \frac{1}{2}(|P| - r_{t-n}) \tag{23}$$

In the above equation, t represents the target order of repair, and r_t represents the number of paths to be abandoned to repair the nodes of order t. These paths contain nodes of the target order. Because abandoning the path will affect the properties of low-order nodes, that is, security and availability, we need to use n to represent the order difference from the target order to the lower order to describe the properties of low-order nodes. If $n = 0$ is taken in, r_t can be obtained, and then the following simple general formula can be obtained by combining Eq. (23).

$$r_{t-n} > 2t - |P| - n \tag{24}$$

The more abandoned paths there are, the more low-order nodes will be affected, and the abandoned path algorithm is more difficult to determine, so we can take the minimum integer in the range. When we set n as some specific values to ensure that r keeps a nonnegative argument, it is not necessary to abandon the path containing nodes of the $t - n$ order when repairing the nodes of the target order.

As a supplementary algorithm, we only carry out the simplest repair in this experiment, that is, dangerous nodes whose order is close to the critical value. When the number of paths is not too large, the number of dangerous nodes can be used as the evaluation index, and the best solution can be found by traversal.

5. Simulation

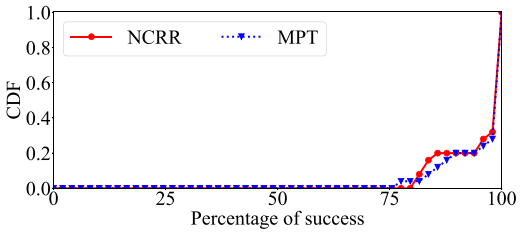
5.1. Simulation setup

In our evaluation, we use two real-world network topologies collected by the Topology Zoo (Knight et al., 2011). The number of nodes and edges of the two topologies is shown in the Table 4. In these topologies, each node is given a unique ID with latitude and longitude. In our simulation, each node is an SDN switch. Any two nodes have a traffic flow, and each flow is forwarded on its selected path. Through SMORE's oblivious path selection method (Kumar et al., 2018), we obtain superior paths that have low stretch for minimizing latency, high diversity for ensuring robustness, and good overhead balancing for achieving performance. The set of these paths is used as the input of the algorithm, and a maximum of ten paths is set. In our simulation, we let the value of K be obtained randomly between 20 and 50 and the value of N be obtained according to the corresponding algorithm, and take their quotient $\frac{N}{K}$ as the overhead ratio. The overhead is determined by the path set output by the algorithm. Finally, we compare the number of packets per flow on each node with the K value of this flow to detect which goal has been achieved.

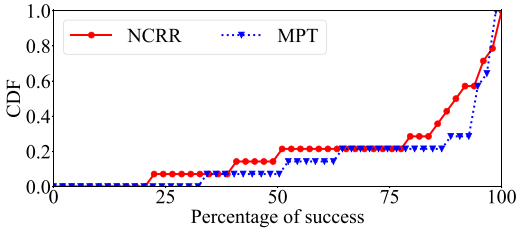
To realize transparent transmission for sender and receiver, our paper is based on Network Function Virtualization (NFV) as shown in Fig. 1. In NFV, virtualized network functions are usually realized by programs in virtual machines or physical servers. Thus, our proposed NCRR is deployed in NFV servers that connect to edge switches. Alternatively, we can also deploy NCRR in the SDN controller. For this case, we need to match the input and output of NCRR to the selected APIs of a controller and develop a specific APP for the controller. Based on the authors' industry experience, different controllers have different design structures and APIs, and it is a time-consuming job. Our main goal is to verify the functionality of the proposed solution. In the future, we consider deploying NCRR for different types of controllers.

5.2. Comparison algorithms

(1) MultiPath-Tree (MPT) (Mohan et al., 2018): this algorithm first transforms the path set into a tree whose destination is the root node, then allocates packets of the destination according to the security of the node, and finally obtains the number of packets that each path should allocate.

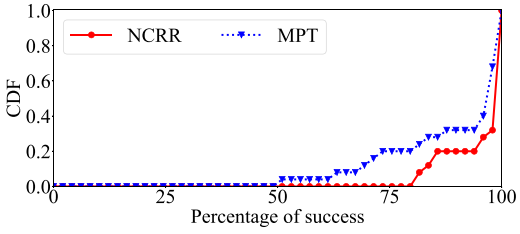


(a) AttMpls.

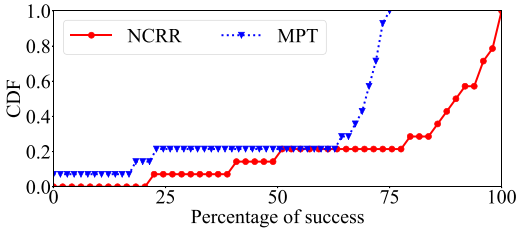


(b) Cernet.

Fig. 5. Data security performance.



(a) AttMpls.

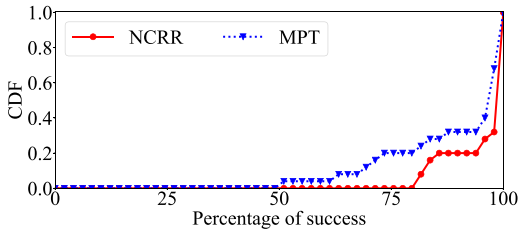


(b) Cernet.

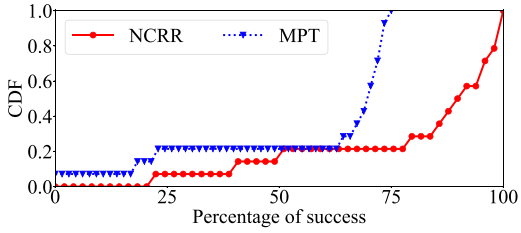
Fig. 6. Data availability performance.

(2) NCRR: this is our proposed algorithm that has been discussed in detail in Section 4.

Both MPT and NCRR consider data security and availability. In the MPT algorithm, ensuring security is taken as a constraint, and maximizing availability is taken as the objective function. In our algorithm, ensuring data security and availability are both taken as constraints, and the objective function is to minimize the total length of paths for forwarding segments. Therefore, NCRR and MPT share similar design concerns and can be compared with each other.



(a) AttMpls.



(b) Cernet.

Fig. 7. Joint data security and availability performance.

5.3. Simulation results

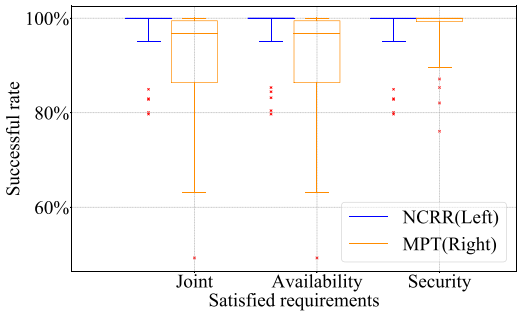
We analyze the results from two perspectives: joint data security and availability, and routing overhead. Realizing joint data security and availability is our main target. Specifically, data security denotes that the security requirement ensures that an attacker cannot decode the flow. Data availability denotes that the requirement ensures that the destination server has the original segments of a flow to recover data. Routing overhead is the ratio of the number of NC segments after redundancy to that before redundancy and shows the cost of reaching data security and availability together.

Figs. 5, 6, and 7 show the simulation results of the CDF diagram on data security, data availability, and joint data security and availability, respectively. Figs. 8 and 9 depict the box-plot of security plus availability and overhead results, whereas Fig. 10 shows the histogram of success rate at specific nodes in the topology.

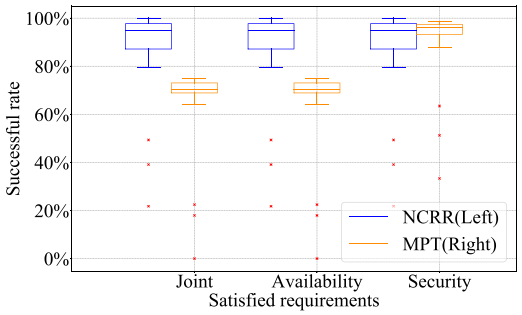
5.3.1. Data security and availability performance

In this point of the section, we discuss the data security and data availability performance separately, first; followed by a detailed discussion on the balance of security and availability performance. The mentioned simulation results and figures are further discussed here to emphasize on joint data security and availability performance.

Data security performance. Fig. 5 shows that, compared with MPT, NCRR has slightly worse security performance in AttMpls and Cernet topologies. It is worth noting that both AttMpls and Cernet topologies have a small number of nodes, which have a high indegree. The small number of nodes leads to the small number of paths that can be used for screening; The number of high indegree nodes in the path leads to more dangerous nodes when using Algorithm 4 of NCRR. This affects the security performance of NCRR to a great extent. From the perspective of the design principle, the unstable part of NCRR is the scenario of Algorithm 4. Because of the principle of evenly distributing packets, this makes all nodes in the path with indegree greater than half the size of the path set dangerous. However, in this case, when the size of the path set is large, part of the dangerous nodes will become safe due to the significant repair effect of PSOP in Algorithm 4. For MPT, the security performance depends on the outdegree of the root node. When the outdegree of the root node is high, the number of packets

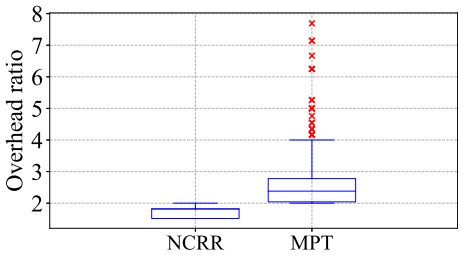


(a) AttMpls.

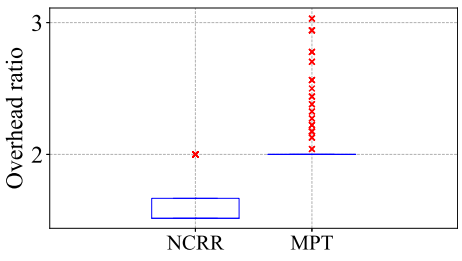


(b) Cernet.

Fig. 8. Joint data security and availability performance.



(a) AttMpls.



(b) Cernet.

Fig. 9. Overhead performance. It shows the ratio of N to K.

Table 4
Topology information.

Topology	Number of nodes	Number of links
AttMpls	25	57
Cernet	14	32

Table 5
Pathset.

Index	Path
1	[0,13,12]
2	[0,11,5,12]
3	[0,9,5,12]

on the children will become a dangerous threshold, and other nodes, such as children or branches of other nodes, are generally more secure. According to the experimental results, the security performance of MPT is better in these two topologies.

Data availability performance. Fig. 6 describes that the availability performance of NCRR is significantly better than that of MPT on these two topologies. In terms of design principles, similarly, the uncertain part of NCRR is in Algorithm 4. Due to the principle of evenly distributing packets and to recover data as much as possible, we choose to protect nodes whose indegrees are at most half the size of the path set. This makes the availability of information affected only when individual nodes with high indegree are attacked. MPT mainly pays attention to the number and connection of child nodes of the root node, and pays less attention to other nodes. Therefore, in terms of data availability, the overall performance of NCRR is better.

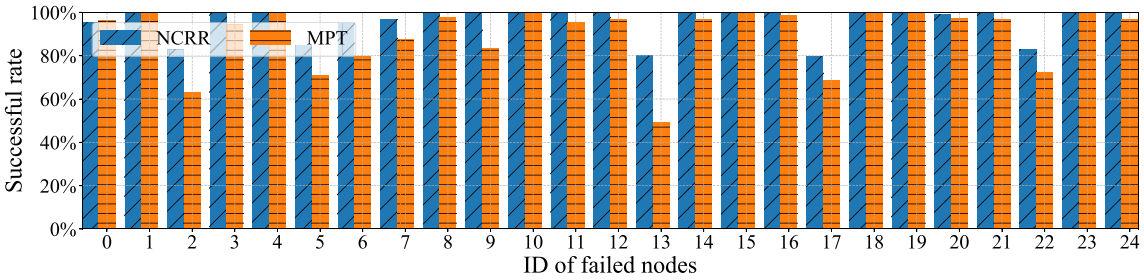
Data security and availability performance. In Fig. 7, it is not difficult to see that the NCRR performs better than the MPT when data security and availability are implemented simultaneously. Although these two algorithms have their own design characteristics, they obviously obey Liebig's law[1] in achieving data security and availability at the same time. From a macro point of view, the MPT mainly considers data security requirements in its design, while the NCRR considers joint requirements. Compared with MPT, NCRR needs to sacrifice some data security performance to achieve better data availability requirements.

Joint data security and availability performance is decided by data security performance and data availability performance, whichever is lower. The simulation results show that joint data security and availability performance of NCRR can be improved if we can improve the data availability performance since the data availability performance is usually worse than data security performance.

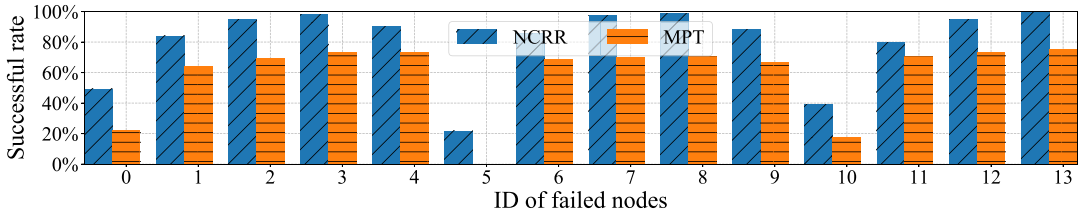
[1] https://en.wikipedia.org/wiki/Liebig%27s_law_of_the_minimum(accessed 6 June 2021).

In the NCRR there are three different situations. The algorithms 1, and 3 can achieve data security and availability, which are the ideal situation. Algorithm 4 can only try its best to achieve data security and availability. In the design of the algorithm 4, for thresholds, we prefer security performance as seen from the rounding of d at the threshold, so data security performance will be slightly better than data availability performance. However, due to the influence of PSOP in Algorithm 4, the nodes at the critical point of safety and danger often become safe, while the remaining dangerous nodes usually have too high indegree, so joint data security and availability are not implemented. Thus, security performance and availability performance tend to be similar.

Here, we provide an example to illustrate why NCRR is better. For the path set in Table 5, the tree in the MPT algorithm is constructed, as shown in Fig. 4. We assume that the initial data has K segments. In the MPT algorithm, there are N segments after redundancy, $N = 2K$. In the experiment, it is assumed that all nodes may be attacked the same, so K segments are transmitted on node 13 and node 5, which means that the MPT algorithm will have two dangerous nodes for such a situation. In the NCRR algorithm, this occasion will be solved by algorithm 4. Whether a node is safe depends only on the size of the indegree and half the size of the path set. In this example, the node with the highest indegree is node 5, which is contained by two paths. Because it is larger



(a) Details of implementing joint data security and availability on each node. Topology: AttMpls.



(b) Details of implementing joint data security and availability on each node. Topology: Cernet.

Fig. 10. Results of each node in topology.

than the defense ability $d = \lfloor \frac{|P|}{2} \rfloor = \lfloor \frac{3}{2} \rfloor = 1$, node 5 is dangerous. From the perspective of the number of segments, the number of segments transmitted on each path is $\lceil \frac{K}{|P|-d} \rceil = \frac{K}{2}$. Only node 5 has K transmitted segments; that is, only node 5 is a dangerous node, which shows that the number of dangerous is less than that of MPT.

5.3.2. Overhead

In Fig. 9, on the whole, the overhead ratio of the NCRR algorithm is lower than that of the MPT algorithm, which means that the overhead pressure in the NCRR algorithm is smaller than that of the MPT algorithm. Because of the rounding problem with the actual value, the actual overhead ratio will float around the theoretical value. In the NCRR algorithm, the theoretical overhead ratios of the four scenarios are 1.50, [1.67, 2.50], 2.50 and 2.0. In the MPT algorithm, the theoretical overhead ratio is related to the number of paths. For each flow, the overhead ratio R is in the range of $[2, num_{root}]$, except for special cases, where num_{root} is the number of child nodes of the root node. According to statistics, in the NCRR algorithm, flows are implemented by Algorithm 1 and 4 account for more than 99% of the total, so the overall overhead ratio will be lower than that of MPT algorithm.

We see that there are few cases where the overhead ratio is 1 because sometimes all paths of P are connected to the destination node through a common node. However, this kind of case is very rare, so it cannot raise the overall redundancy barely.

6. Related works

With the deepening of information technology, people pay increasing attention to the operational security of the system and the protection of information privacy. The security and availability of information have witnessed unprecedented attention.

6.1. Security

Stavrou and Pitsillides (2010) and Zin et al. (2015) discuss the different network design objectives, and classify the existing WAN routing algorithms along with their advantages and disadvantages. Their research discusses that the security of information is getting unprecedentedly continuous attention, especially in some critical information transmission (such as military communication). Yazdinejad et al. (2020) propose a model with multiobjective optimization to find the optimal number of replicated devices, and they realize this model using the NSGA-II (Deb et al., 2002) algorithm. In their proposed method, the effect on security routing has a tight bond with the number of replicated switches or routers. The more replicated devices there are, the greater the possibility of security routing, also leading to higher hardware and management costs in the SDN. Zhang et al. (2019) propose a Lagrangian Relaxation based Bellman–Ford Parallel algorithm (LRBFP) algorithm to ensure security routing when a disaster occurs by stopping the GSLB device from broadcasting the IP of the disaster-stricken data center and transferring incoming flow using other paths calculated by a normal data center. This is a kind of active defense and may lead to abrupt calculation increases in normal data centers and sudden higher bandwidth requirements, which can even lead to possible segments being lost and affect routing in other normal areas. The above-mentioned two algorithms either add devices to the SDN network or bring possible abrupt flow increases. They are both data security-centric schemes.

6.2. Availability

Zhang et al. (2015) propose a Generalized Destination-based Multipath Routing (GDMR), which achieves the same high performance as display routing and achieves better overhead balancing in IP networks. Yan et al. (2015) propose a HiQoS algorithm, which uses SDN to provide QoS guarantees and uses multipaths and queuing mechanisms to make QoS more comprehensive. In this algorithm, the flow on the fault route will be rerouted to other available paths to achieve better availability. Sheu et al. (2016) propose an efficient heuristic algorithm for k Max-Min bandwidth disjoint paths. It can significantly improve communication throughput by finding K paths with the largest bottleneck bandwidth. These three algorithms balance the overhead through multiple paths to achieve higher availability and improve the efficiency of communication. The algorithms mentioned here are data availability-centric solutions. However, we need excellent performance in joint security and availability because both are equally important in some scenarios.

6.3. Security and availability

Mohan et al. (2018) develop a heuristic solution MultiPath-Trees (MPT) for Attack Resilient Routing (ARR) by using a multipath tree, which can guarantee the security of data and maximize availability as the goal in the presence of possible attacks. The algorithm proposed by Doshi and Kamdar (2018) makes the core network overhead balance effective, reduces congestion, and improves reliability. They compute k-max min disjoint paths and use an analytic hierarchy process to sustain varying quality of service requirements and maintain the quality of service subject to multiple criteria. These two algorithms consider joint security and availability, but unfortunately, the MPT algorithm cannot achieve a good balance in data security and availability; for the algorithms in Doshi and Kamdar (2018), they require that multi-paths should be disjoint paths, which is difficult to achieve in some practical situations. Therefore, we propose the NCRR algorithm, which balances the data security and availability, and at the cost of relatively low redundancy, does not need to do disjoint rigid requirements for multipath, and it can be applied to all networks supporting SDN.

7. Conclusion

In this paper, we proposed a secure routing algorithm called NCRR to jointly realize data security and availability in SDN. NCRR works for three specific scenarios based on the number of disjoint paths. For each scenario, NCRR selects forwarding paths and decides the number of segments on the selected paths for each flow to achieve jointly data security and availability in SDN. The extensive evaluation shows that NCRR can achieve secure routing and effectively resist attacks for large-scale networks. In the future, we will extend our work from three aspects. First, we can design an efficient heuristic algorithm to improve joint data security and availability performance. Second, we can extend NCRR's application scenarios by deploying it for different types of controllers. Third, we can use a flexible objective that a user can customize the priority of data security and data availability.

CRediT authorship contribution statement

Haoran Ni: Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Zehua Guo:** Conceptualization, Methodology, Resources, Writing – original draft, review & editing, Supervision, Project administration. **Changlin Li:** Writing – review & editing, Visualization. **Songshi Dou:** Writing – original draft, Visualization. **Chao Yao:** Writing – review & editing. **Thar Baker:** Writing – review & editing.

Declaration of competing interest

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

Acknowledgments

This work was supported in part Natural Science Foundation of China under Grant 62002019, Beijing Institute of Technology Research Fund Program for Young Scholars, and Fundamental Research Funds for the Central Universities under Grant 1301032207.

References

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197.

Doshi, M., Kamdar, A., 2018. Multi-constraint QoS disjoint multipath routing in SDN. In: 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT). pp. 1–5.

Juan, G., Pedersen, M., Fitzek, F., 2021. Network Coding. pp. 169–195.

Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M., 2011. The internet topology zoo. IEEE J. Sel. Areas Commun. 29 (9), 1765–1775.

Kumar, P., Yuan, Y., Yu, C., Foster, N., Kleinberg, R., Lapukhov, P., Lim, C.L., Soulé, R., 2018. Semi-oblivious traffic engineering: The road not taken. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). USENIX Association, Renton, WA, pp. 157–170, [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/kumar.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 38 (2), 69–74.

Mohan, P.M., Gurusamy, M., Lim, T.J., 2018. Dynamic attack-resilient routing in software defined networks. IEEE Trans. Netw. Serv. Manag..

Rischke, J., Salah, H., 2021. Software-Defined Networks. pp. 107–118.

Sathiamoorthy, M., Asteris, M., Papailiopoulos, D., Dimakis, A.G., Vadali, R., Chen, S., Borthakur, D., 2013. XORing elephants: Novel erasure codes for big data. Proc. VLDB Endow. 6 (5), 325–336, [Online]. Available: https://doi.org/10.14778/2535573.2488339.

Scott-Hayward, S., O'Callaghan, G., Sezer, S., 2013. Sdn security: A survey. In: 2013 IEEE SDN for Future Networks and Services (SDN4FNS). pp. 1–7.

Sheu, J.P., Liu, L.W., Jagadeesha, R.B., Chang, Y.C., 2016. An efficient multipath routing algorithm for multipath TCP in software-defined networks. In: European Conference on Networks and Communications.

Stavrou, E., Pitsillides, A., 2010. A survey on secure multipath routing protocols in WSNs. Comput. Netw. 54 (13), 2215–2238.

Yan, J., Zhang, H., Shuai, Q., Liu, B., Guo, X., 2015. HiQoS: An SDN-based multipath QoS solution. China Commun. 12 (5), 123–133.

Yazdinejad, A., Parizi, R.M., Dehghantanha, A., Srivastava, G., Mohan, S., Rababah, A.M., 2020. Cost optimization of secure routing with untrusted devices in software defined networking. J. Parallel Distrib. Comput. 143, 36–46, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731519306288.

Zhang, W., Ma, L., Jiang, X., 2019. Disaster-aware dynamic routing for SDN-based active-active data center networks. In: 2019 International Conference on Networking and Network Applications (NaNA). pp. 160–165.

Zhang, J., Xi, K., Chao, H.J., 2015. Load balancing in IP networks using generalized destination-based multipath routing. IEEE/ACM Trans. Netw. 23 (6), 1959–1969.

Zin, S.M., Anuar, N.B., Kiah, M.L.M.M., Ahmedy, I., 2015. Survey of secure multipath routing protocols for WSNs. J. Netw. Comput. Appl. 55, 123–153.

Haoran Ni is currently pursuing B.S. degree of Electrical Engineering at Beijing Institute of Technology, Beijing, China. His research interests cover software-defined networking and routing algorithm.

Zehua Guo received B.S. degree from Northwestern Polytechnical University, Xi'an, China, M.S. degree from Xidian University, Xi'an, China, and Ph.D. degree from Northwestern Polytechnical University. He was a Research Fellow at the Department of Electrical and Computer Engineering, New York University Tandon School of Engineering, New York, NY, USA, and a Research Associate at the Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA. His research interests include programmable networks (e.g., software-defined networking, network function virtualization), machine learning, and network security. Dr. Guo is an Associate Editor for IEEE Systems Journal and the EURASIP Journal on Wireless Communications and Networking (Springer), and an Editor for the KSII Transactions on Internet and Information Systems. He is serving as the TPC of several journals and conferences (e.g., Elsevier Computer Communications, AAAI, IWQoS, ICC, ICCCN, ICA3PP). He is a Senior Member of IEEE, China Computer Federation, China Institute of Communications, and Chinese Institute of Electronics, and a Member of ACM.

Changlin Li received B.S. degree from the School of Information Science and Engineering, China University of Petroleum, Beijing, China, in 2020. He is currently pursuing M.S. degree at Beijing Institute of Technology, Beijing, China. His research interests cover software-defined networking, traffic engineering and network optimization.

Songshi Dou received B.S. degree from the School of Control and Computer Engineering, North China Electric Power University, Beijing, China, in 2019. He is currently pursuing M.S. degree at Beijing Institute of Technology, Beijing, China. His research interests cover software-defined networking, traffic engineering and network optimization.

Chao Yao received his B.Sc. in telecommunications engineering in 2007, and the Ph.D. degree in communication and information systems in 2014, both from Xidian University, Xi'an, China. He was a visiting student in Center for Pattern Recognition and Machine Intelligence (CENPARMI), Montreal, Canada, during 2010–2011. His research interests include feature extraction, handwritten character recognition, machine learning, and pattern recognition.

Thar Baker received the Ph.D. degree in autonomic cloud applications from Liverpool John Moores University (LJMU), Liverpool, U.K., in 2010. In 2018, he became a Senior Fellow of Higher Education Academy. He has authored/coauthored numerous refereed research papers in multidisciplinary research areas including big data, algorithm design, green and sustainable computing, and energy routing protocols. Dr. Baker has been actively involved as a member of editorial board and review committee for a number of peer-reviewed international journals, and is on program committee for a number of international conferences. He is an Associate Editor for Future Generation Computer Systems. He is an Expert Evaluator of European Horizon 2020 (EU H2020), Information and Communication Technology (ICT) Fund, and British Council.