Tchendji et al. Journal of Internet Services and Applications (2018) 9:13
https://doi.org/10.1186/s13174-018-0085-4

Journal of Internet Services and Applications

RESEARCH Open Access

Conflict-free rerouting scheme through flow splitting for virtual networks using switches

Vianney Kengne Tchendji[1], Yannick Florian Yankam[1] and Jean Frédéric Myoupo[2*]

Abstract

The weaknesses of the Internet led to the creation of a new network paradigm – network virtualization. Virtualization is a very successful technique for sharing and reusing resources, which results in higher efficiency. Despite its advantages, including flexibility in network architecture, virtualization imposes many challenges, such as physical resource allocation to virtual devices. An efficient allocation strategy for these resources can ensure good Quality of Service (QoS) in virtual networks, whether in node or link failure events. This paper presents a conflict-free rerouting scheme with efficient additional capacity usage for link and node failure resilience in a virtual network using switches. Combining an IP Fast Rerouting approach and flow-splitting strategy, this scheme provides short reaction time, stable performance and low complexity because the rerouting calculation and configuration are performed in advance. We show that rerouting by traffic splitting based on the entering arc and destination is sufficient to address all link-failure situations in the network, assuming that the network is two-link connected. After modelling the dimensioning problem as an Integer Linear Programme, we demonstrate through practical implementation of our rerouting scheme on different networks that the scheme can substantially minimize the additional capacity draw on the substrate network. A solution using multiple virtual planes is also provided to solve several conflict problems in the case of simultaneous multiple link failures.

Keywords: Network virtualization, Quality of service (QoS), Restoration scheme, Spare capacity, Traffic splitting

1 Introduction

Since its creation, the Internet has brought innovations and success to industry, economic and research fields; however, deployment of any new, radically different technology and architecture is becoming highly difficult, a situation that cloud computing can mitigate. That effect is what we call Internet ossification [1]. To fend off ossification, studies have proposed rethinking the architecture of the actual Internet [1]. However, network virtualization is the most promising approach to addressing current limitations of the Internet and supporting new requirements [2–5]. Its principle is to implement multiple virtual routers on each physical machine and to interconnect them through substrate network architecture. This implementation allows virtual networks to have different logical topologies from that substrate network, and each of them behaves as a true network in which it is possible to implement different routing protocols and services. As in the substrate network, failures could occur in virtual networks; in this case, rerouting mechanisms can be implemented to forward traffic by using available resources in the virtual network or additional ones taken from the physical network. This additional resource could cause dysfunctional risks inside another virtual plane.

1.1 Previous work

The link and node failure problem has been investigated for a long time in the framework of physical networks but not in virtualized networks because of the new requirements brought by this technology [6] (e.g., architecture flexibility, mobility, and isolation). Our restoration scheme is pre-calculated. Therefore, the rerouting paths for all link failure cases are determined and recorded inside the controller in advance. When a failure

* Correspondence: jean-frederic.myoupo@u-picardie.fr
[2]Computer Science Lab-Mis, University of Picardie Jules Verne, Amiens, France
Full list of author information is available at the end of the article

occurs, the nodes apply the pre-calculated rerouting paths directly. Multiple methods based on the IPFRR (IP Fast Reroute) strategy have been proposed for transient failures. However, they have the following limitations:

- For rerouting schemes using a single path and additional capacity [7, 8], the limits of the physical resources are quickly reached, which paralyses the network.
- For loop-free alternative mechanism-based methods [9], we are not certain of rerouting traffic to all destinations; doing so only helps to reduce the number of lost packets in an IP network.
- Not-via addressing [10] and tunnelling [11] mechanisms require encapsulation and de-encapsulation of packets, whereas in a multiple routing configuration mechanism [12], the packets must carry configuration information. With the appearance of optic networks, methods that modify the packets are not recommended but can instead be used to optimize the usage of resources in network virtualization.
- Multipath rerouting [13] using spare capacity in the network can induce a capacity saving of up to 11% in randomly generated networks, but lack of spare capacity due to the existence of multiple virtual planes on a substrate network can undermine this result in network virtualization.

Additionally, [14–16] propose methods to find back-up paths that permit rerouting traffic in the case of link failure but not in the context of network virtualization. Proposed schemes also based on IPFRR use two port types: primary and back-up ports. The traffic will change from the primary port to a back-up port only when there is a failure on the primary port or the traffic comes from the primary port. This packet forwarding strategy uses a bridge to reconnect sub-graphs coming from a failure but does not consider conflicts, which can disturb traffic. In contrast to these works, our rerouting scheme uses not just one but multiple bridges, and it avoids conflict on rerouting paths. Our strategy extends the results in [7] and can minimize the dimensioning cost of the network. Recently, a restoration scheme was proposed in [7, 8] to handle both node and link failure problems by replacing all or parts of a network with switches managed by an external controller. The authors summarized the previous works [17–20] and noted their drawbacks. They derived a rerouting scheme negating these drawbacks. Their rerouting model uses only one path to reroute traffic and to minimize the additional capacity used by a virtual network; it cannot solve the multiple-link-failure problem. Moreover, their work in [7, 8] has drawbacks:

- Network modelling using only one rerouting path cannot significantly minimize additional capacity, which means that physical resource limits are quickly reached and the network is paralysed for a moment. Because the physical resource limits are quickly reached in [7, 8], the number of failures handled is also reduced.
- Based on their rerouting model, the authors in [7, 8] claim that a conflict-free rerouting scheme for the multiple-link-failure situation cannot be constructed even when the network remains connected.

The work in [21] presents a virtual network embedding model that allows a virtual link to map multiple substrate paths. This model can help to build a rerouting strategy using multiple planes.

1.2 Our contribution

We initially propose a new scheme to solve the link and node failure problems in a network of switches that avoids conflict on rerouting paths in contrast to protocols in [6, 7]. The new rerouting scheme we present here clearly uses not just one but multiple bridges, and it avoids conflict on rerouting paths. Our strategy uses a flow-splitting technique [22–25], extends the results in [7, 8] and can minimize the dimensioning cost of the network. Flow splitting is a method for restoring traffic from a failed link using multiple rerouting paths in the case of insufficient residual capacity. In this first contribution, we consider only one virtual plane. We propose a rerouting scheme that ensures that for any link or node failure, the traffic will be rerouted until it reaches the destination through an alternative path.

Given that there generally are no rerouting solutions that avoid conflicts in all network configurations, our second contribution is to avoid these situations. To reach this goal we use multiple planes to provide a rerouting strategy that avoids conflicts that could not be solved by using only one plane. Therefore, we introduce here a new scheme to solve the link and node failure problems in a network of switches. This new scheme negates the drawbacks of [7, 8] by showing that a conflict-free rerouting scheme for a multiple-link-failure situation can be constructed even when the network remains connected. The rerouting scheme described in this paper uses filters in switches to determine the next hop for the incoming flow. We provide each virtual network a specific filter. Then, the controller sets the flow path by programming the switches in the form of quadruplets (S, N, O, F) in which S is the source port (node), N the current node, O the output ports because flow can be split and forwarded through different paths to the same destination depending upon the spare capacity needed, and F the filter, which indicates the destination.

More precisely, for an incoming flow from a neighbour and a given destination, the scheme will assign the potential output ports. In the case of failure, only the upstream node must react by directing the disturbed traffic to one or many of its neighbours. The traffic is routed according to the filter programmed in each node of the network. Traffic can be split anytime at the level of each node if needed. Hence, the proposed scheme needs only a local reaction, making its implementation particularly easy in distributed environments. This local reaction helps the network operate normally and can solve the problem of transient failures. A transient failure is a failure whose duration is short, less than 10 min, whereas the duration of a persistent failure is longer [7]. When the failure is determined to be persistent, the controller can recalculate the routing table for all nodes in the network. To avoid the rerouted traffic of a failure causing disturbances to another part of network, additional capacity is added to all arcs. Because this additional capacity is added in the physical network and is exploited by several virtual layers, it is necessary to minimize it. The mathematical model that we propose in this paper can calculate the rerouting paths and optimize the total additional capacity injected into the network.

To the best of our knowledge, presently our work is the only one that shows how to effectively solve simultaneous multilink failures using flow splitting methods, thus providing an improvement in QoS of computer networks.

The rest of this paper is organized as follows. The next section presents motivations for traffic splitting. Section 3 provides a full description of our restoration scheme for a link failure configuration. Our mathematical model is described in Section 4. Section 5 presents numerical results of implementations. Section 6 studies the application of our rerouting scheme to the single-node-failure problem. Section 7 extends our work to simultaneous multiple-link-failure situations by providing a solution for some conflict problems. Finally, Section 8 ends the paper.

2 Motivation for traffic splitting

2.1 Improvements of computer networks Qos

Virtual networks use physical network resources to achieve their needs. In the case of link or node failure, spare capacity available in safe links is commonly used to restore traffic. However, this spare capacity might not be sufficient to carry the entering traffic; this situation represents a lack of spare capacity and is the main motivation for using flow splitting in the network. The idea is to split an original flow into multiple parts such that they can be forwarded easily through the network. This method induces numerous potential advantages as:

- Reduction of transit delay and packet loss rate, because the flows are more able to reach their destination nodes, thus improving the network QoS;
- improvement of the packets routing delays: since the original flow is split into several lighter-sized streams, they can be transported more quickly to the destination;
- improvement of load balancing distribution [26], leading to prevent or decrease congestion risk across the network. This is a well-known benefit of flow splitting in computer networks.
- extension of the lifetime of a network by allowing more flexible and efficient resources allocation;
- better economy of the substrate network resources supporting virtual networks. Since virtual networks are built on a physical network infrastructure, it is necessary to avoid an abuse of these resources with the risk of causing the hosted virtual networks malfunctions.

We illustrate this last point by an example. Figure 1 shows a network with 6 nodes and 8 links. The numbers carried by each arc represent spare capacity available on the links. There is only one path (of traffic) 6–5–2-1 going through link (5, 2). Other paths are not shown for clarity. Let the traffic of this path be 8 units, and let all links be of equal length. In link restoration, a faulty link is replaced by one alternative path.

Assume that faulty link (5, 2) is replaced by path 5–3-1. Because links (5, 3) and (3, 1) have only 5 and 7 spare capacity each, both links need, respectively, 3 and 1 more spare capacity to make the restoration possible. This example proves that the use of traffic splitting in the link restoration scheme can result in lower spare capacity requirements but in the context of a virtual network, it could be very important to reduce additional capacities added to the links to avoid disturbances due to the rerouting scheme.

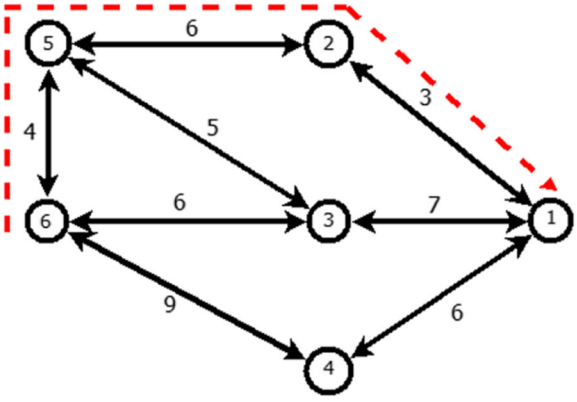

Fig. 1 Example network

Now, let us consider the traffic-splitting version of link restoration (Fig. 2). There are two alternative paths, 6–5–3-1 and 6–3-1, for link (5–2), and each alternative carries 4 units of traffic. (As in the model [22], the general design principles presented in this paper are valid for any unit of bandwidth capacity for virtual networks.) With this second method, there is no need for more spare capacity.

2.2 The packets re-ordering problem

Beyond its advantages, flow splitting also brings some difficulties, the best known of which are the following:

- Avoiding the packet re-ordering. When a stream is subdivided into the network, the different parts must be reassembled without losing no part (a potential TCP performance problem [27, 28]), making the use of flow splitting strategy very delicate. So, we should try to make reordering rare. Therefore avoiding this reordering until the packets reached the destination is crucial. However to overcome this problems some approaches were elaborated. Some strategies separate traffic at the level of flows. This approach removes the problem of reordering but at the cost of a restriction in the granularity with which we can split traffic [29]. Another one operates on bursts of packets (flowlets) carefully chosen to avoid reordering, but allowing a finer granularity of load balancing [30]. Some other algorithms [31, 32] minimize or eliminate reordering in some situations. But, some reordering problem should occur, and probably often enough to affect performance of IP networks;
- the problem of reassembling packet segments inside the destination node. Due to various reasons, such as multipath routing, route fluttering, and retransmissions, packets belonging to the same flow may arrive out of order at a destination. The problem is how to know which packet comes before or after another one when we want to rebuild the original flow packets [27–29]. Some algorithms based on packet numbering in [29] can be used to at least minimize reordering.

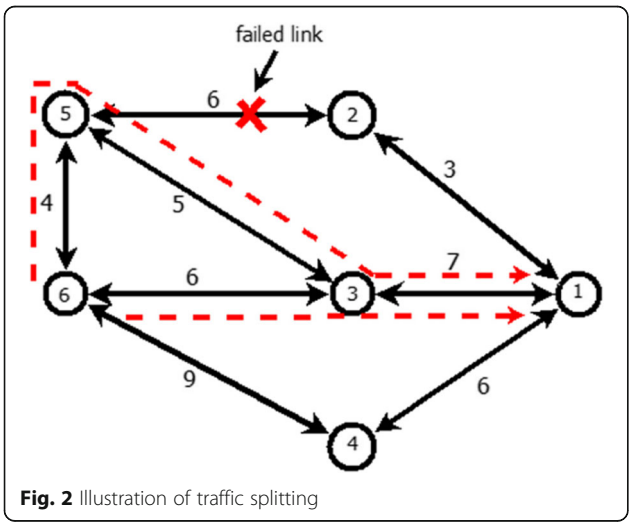

Fig. 2 Illustration of traffic splitting

In this work, flow splitting is implemented by building little flow of packets from an original one. To face reordering challenge, we use the numbering packets each time the flows are split, because this method does not modify significantly the packets headers.

3 New multipath link failure restoration scheme

In this section, we present our rerouting scheme that addresses the case of a link failure. As presented in Section 2, traffic splitting occurs because spare capacity is lacking in the network, but implementation of that approach in our rerouting scheme helps to solve many other problems:

- Safeguarding of network resources by minimizing spare and additional capacity usage to manage more traffic
- Possible rerouting, however, is impossible to do with only one path, as shown in [7].

A routing tree called a nominal routing tree is associated with a given destination; this tree is constructed using the shortest path tree criterion. We assume that the routing is provided. In the case of a failure of an arc or edge (both arcs are then concerned) and a lack of spare capacity in links, we reroute the traffic through one or more alternative paths. When there is only one path used, our rerouting scheme is similar to [7]. According to the routing scheme, for two independent failures, if two rerouting paths to a given destination have a common arc, they must merge after this arc. This requirement holds for both nominal and rerouting paths. If two paths do not satisfy this requirement, we say that they are in conflict. Any routing scheme satisfying this requirement is said to be without conflict. In this strategy, only the extremity failed link nodes will know about the failure. The upstream nodes initiate the traffic diversion, whereas all other nodes in the network apply the filter for each incoming flow without any difference between disturbed and non-disturbed flows. Because the disturbed traffic is rerouted on multiple alternative paths and should satisfy the non-conflict requirement, the cost in terms of resources and of computational time is expected to be higher compared with conventional schemes using single path rerouting.

We illustrate this rerouting scheme in Figs. 3 and 4.

These figures represent a network with 6 nodes and 8 links.

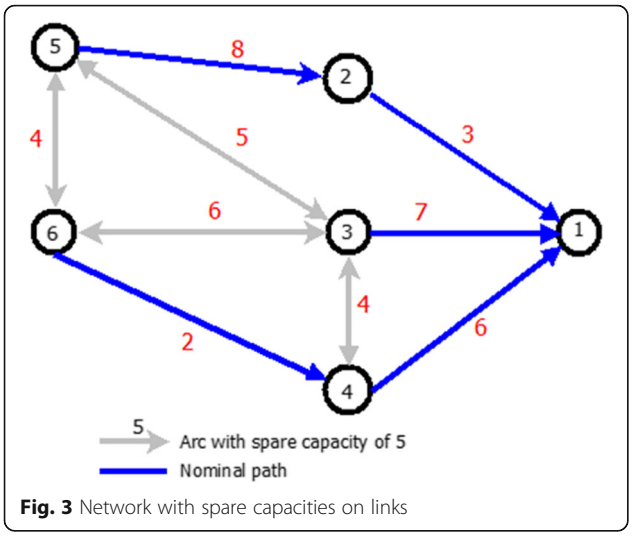

Fig. 3 Network with spare capacities on links

The original graph with thespare capacity of each link is shown in Fig. 3.

Assume that the source node is node 6 and that the destination node is node 1. We also suppose that the failure link 4–1 generates a flow weight of 15 units to reroute. Figure 4 represents each link's weight and the additional capacity required on each arc if rerouting paths use this arc. If we reroute using only one alternative path, similar to [7], the path 4–6–3-1 will be selected as the rerouting path (see Fig. 5), which consumes $T = 13 + 9 + 8 = 30$ units of additional capacity.

When applying our rerouting scheme (Fig. 6), part 4–6 of the previous rerouting path will be preserved. From node 2, traffic can be split into two parts because there are two arcs leaving node 6 to node 1. Our splitting criterion is the amount of spare capacity available on links. Thus, a node will send some flow on the link that offers the greatest spare capacity, and the remaining flow will be sent on the other link. Therefore, 6 units of traffic

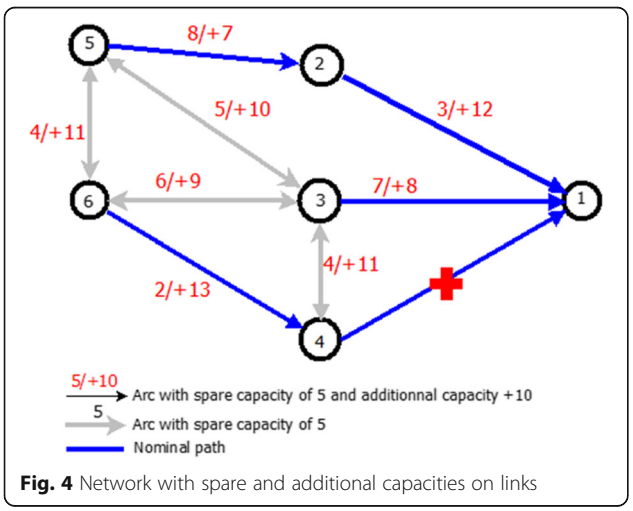

Fig. 4 Network with spare and additional capacities on links

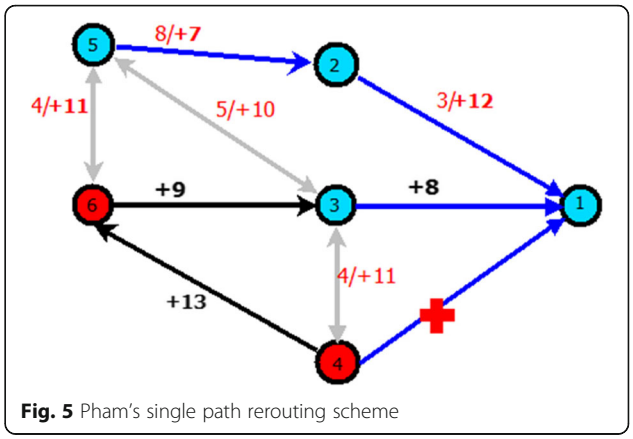

Fig. 5 Pham's single path rerouting scheme

will be sent on arc $(6 \rightarrow 3)$ (which is a bridge), offering a spare capacity of 6, and 9 will be sent on arc $(6 \rightarrow 5)$ (another bridge), which offers a spare capacity of 4. In this example, we use an additional capacity of 5 on arc $(6 \rightarrow 5)$ in addition to its existing spare capacity. The rerouting paths are 4–6–3-1 using $13 + 0 + 0 = 13$ additional capacity, and 4–6–5-2-1 using $13 + 5 + 1 + 6 = 25$ additional capacity. Arc $(4 \rightarrow 6)$ belongs to the two rerouting paths and involves a total additional capacity usage of 13. Therefore, the total additional capacity needed for these two paths is 25, i.e., a total additional capacity savings of 17% relative to [7]. This analysis shows that our rerouting scheme can substantially minimize the additional capacity needed on links to reroute the traffic of a failed link. Because of the configuration of filters, traffic will be rerouted until it reaches the destination without any conflict.

4 Mathematical formulation

4.1 Description of our model

This section provides a rerouting mathematical model based on the following assumptions:

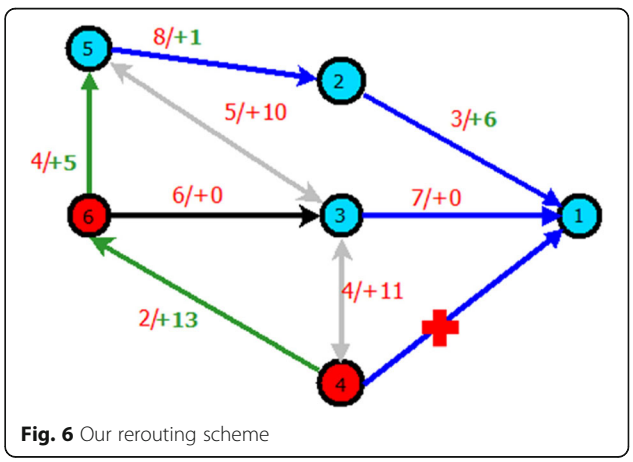

Fig. 6 Our rerouting scheme

- The graph is assumed oriented and symmetric.
- There are at least two disjoint arc paths between any two nodes of the graph.
- There is only one link failure at a time.

To resolve the question of the existence of a rerouting solution without conflict, we have the following theorem:

Theorem 1. *For any destination d, there is a rerouting plan without conflict using one or many alternative paths.*

The formal mathematical proof can be found in the Appendix.

As in [7], a similar mathematical formulation can be provided for our case, but we add in the equations the number of rerouting paths. Consider the following notation:

- Rt^d: set of arcs of the routing tree to destination d
- Ac_{ab}: additional capacity assigned to arc (a, b)
- Tr_n^d: total traffic for d that passes through node n. n is the node that detects the failure. In fact, the failure is characterized by a source n and a destination d because we use the routing tree for nominal routing. For a destination d, the failed arc is the one routing the traffic going to d and coming from n by nominal routing.
- F_i: indicates fictive nodes used to divert traffic in the case of failure. We introduce the fictive nodes F_i that will be used for all failures. For a given failure (n, d), the traffic to d will be rerouted by i paths from F_i to d starting with arc (F_i, n), $i = 1, 2...$
- SRed_n^d: SRed_n^d sub-tree of sink n. Recall that in the case of failure, the tree is divided into two parts, the isolated part, that is the Red part, and the Blue part. Alternative paths will reroute traffic from the Red part to the Blue part.
- SBlue_n^d: sub-tree of sink d, with $Rt^d - SRed_n^d \mathrm{SRed}_n^d \mathrm{SRed}_n^d$
- y_{efgh}^{dn} y_{efgh}^{dn}: this binary variable indicates whether the e^{th} alternative path to destination d for a given failure contains arcs (f, g) and (g, h); node n is the node that detects the failure.
- x_{efgh}^d: x_{efgh}^d this binary variable indicates the rerouting scheme to destination d. It takes value 1 if there is a failure in which the e^{th} alternative path to destination d contains arcs (f, g) and (g, h). Therefore, the variable takes value 1 if there are n and e where $y_{efgh}^{dn} y_{efgh}^{dn}$ is equal to 1.
- α_{eab}^{dn}: α_{eab}^{dn} this binary coefficient equals 1 if arc (a, b) belongs to one of the paths in the nominal routing from n to d except the failed arc.
- *Quadruplet*: All quadruplets (e, f, g, h) where e is the number of an alternative path in a rerouting scheme, f, g, h are nodes of the graph; f can be the fictive node, and (f, g) and (g, h) are two adjacent arcs, with. $f \neq h$.
- *Arc*: All arcs of the graph
- L: Set of links
- N: Set of nodes

The objective is to minimize the sum of additional capacity allocated to each arc; our objective function is provided by (1):

$$min \sum_{(a,b)\in Arc} Ac_{ab} \tag{1}$$

$$\sum_{(n,h_s)_{\in Rt^d}, h \text{ neighbor of } n,\ h=h_1}^{h=h_s} y_{eF_0 n h_s}^{dn} = 1, n\in N, d\in N, s = 1, 2, ..., e = 1, 2, ... \tag{2}$$

$$\sum_{h_s\in neighbor\ of\ g,\ h=h_1}^{h=h_s} x_{efgh}^d \leq 1, (f,g)\in Arc, d\in N, s = 1, 2, ..., e = 1, 2, ... \tag{3}$$

$$y_{efgh}^{dn} = 0, d\in N, n\in N, j\in SRed_n^d, (f,g)\in Rt^d, (e,f,g,h)\in Quadruplet, e = 1, 2, ... \tag{4}$$

$$\sum_{f\in N|(e,f,g_s,h_s)\in Quadruplet}^{h=h_s} y_{efg_s h_s}^{dn} \leq 1, d\in N, n\in N, (g_s,h_s)\in Arc, g_s\in SRed_n^d, h_s\in SRed_n^d, e = 1, 2, ... \tag{5}$$

$$\sum_{f_s\in neighbor\ of\ f}^{h=h_s} y_{ef_s fg}^{dn} = \sum_{h_s\in neighbor\ of\ g}^{h=h_s} y_{efgh_s}^{dn}, d\in N, (f,g)\in Arc, f\neq f_s, f\neq n, g\in N-d, s = 1, 2, ..., e = 1, 2, ... \tag{6}$$

$$\sum_{e=1}^{e=k}\sum_{s=1}^{s=k} y_{engh_s}^{dn} = y_{eF_i ng}^{dn}, d\in N, n\in N, (n,g)\in Arc, k = 1, 2, ..., e = 1, 2, ... \tag{7}$$

$$y_{eghh_1}^{dn} - y_{efgh}^{dn} \geq 0, h\in SBlue_n^d - d, (h_1,h)\in Rt^d, (g,h)\in Rt^d, \forall d\in N, \forall (e,f,g,h)\in Quadruplet, e = 1, 2, ... \tag{8}$$

$$\sum_{n\in N}\sum_{e=1}^{e=k} y_{engh}^{dn} \geq x_{efgh}^{dn} \geq \frac{\sum_{n\in N} y_{engh}^{dn}}{cardinal(N)}, (e,f,g,h)\in Quadruplet, n\in N, k = 1, 2, ..., e = 1, 2, ... \tag{9}$$

$$\sum_{d\in N|(n,m)\in Rt^d} \sum_{f\in neighbor\ of\ g, f\neq h} \sum_{e=1}^{e=k} y_{efgh}^{dn}.Tr_n^d + \sum_{d\in N|(m,n)\in Rt^d} \sum_{f\in neighbor\ of\ g, f\neq h} \sum_{e=1}^{e=k} y_{efgh}^{dn}.Tr_m^d \leq Ac_{gh} + \sum_{d\in N|(n,m)\in Rt^d} \alpha_{egh}^{dn}.Tr_n^d + \sum_{d\in N|(m,n)\in Rt^d} \alpha_{egh}^{dm}.Tr_n^d, \quad (g,h)\in Arc, g\neq n, g\neq m, (n,m)\in L, k=2,3,\ldots, e=1,2,\ldots \tag{10}$$

$$\sum_{d\in N|(n,m)\in Rt^d} \sum_{i=1}^{i=k} y_{eF_inh}^{dn}.Tr_n^d \leq Ac_{nh} + \sum_{d\in N|(n,m)\in Rt^d} \alpha_{nh}^{dn}.Tr_n^d, (m,h)\in Arc, h\neq n, (m,n)\in L, e=1,2,\ldots \tag{11}$$

$$\sum_{d\in N|(m,n)\in Rt^d} \sum_{i=1}^{i=k} y_{eF_imh}^{dn}.Tr_m^d \leq Ac_{nh} + \sum_{d\in N|(n,m)\in Rt^d} \alpha_{mh}^{dm}.Tr_n^d, (m,h)\in Arc, h\neq n, (m,n)\in L, e=1,2,\ldots \tag{12}$$

$$x_{efgh}^d \in \{0,1\}, \forall (e,f,g,h)\in Quadruplet, \forall d\in N, e=1,2,\ldots \tag{13}$$

$$y_{efgh}^d \in \{0,1\}, \forall (e,f,g,h)\in Quadruplet, \forall d\in N, \forall n\in N, e=1,2,\ldots \tag{14}$$

The objective function will allow us to evaluate the ratio between the additional capacity and the installed capacity.

Equation (2) is a constraint implying that there exist multiple paths resulting from flow splitting, which go from n to d for disturbed traffic.

Equation (3) ensures that there is no conflict in the rerouting, i.e., the incoming flows to node n to destination d must follow the same rerouting paths. If we use arcs (i, k_1), (i, k_2), (i, k_3), ... for rerouting to destination d, there is at most one output (k_s, j_s) for each.

To avoid loops and conflict problems, the alternative paths should not contain any arc of nominal routing in the red part of the network. Equation (4) ensures that condition.

Constraint (5) ensures that there will be no loop in the network. For a given destination and a given failure, an alternative path could contain a loop if the flows go from a node with a larger index number to another one with a smaller index. This constraint prohibits this type of problem.

Equations (6), (7), and (8) are the flow constraints for the continuity of the alternatives paths. Equation (6) is the constraint of flow conservation. Referring to (7), the total amount of entering traffic in n is equal to the total outgoing traffic of g; because of flow splitting being used for a given failure and a destination, we could have multiple incoming streams and possibly multiple outgoing streams.

Equation (8) ensures that in the blue part, if a path uses an arc of the nominal routing tree, it must continue until destination d.

Equation (9) is a constraint for the relationship between two rerouting paths that avoids a conflict (see the definitions of variables x and y). Because x and y are binary variables, with the same quadruplet (e, f, g, h) and same destination d, we can deduce from (9) that (x) will take the maximum value of (y). We use the sum of failures divided by the cardinal to reduce the number of constraints.

Equations (10), (11), and (12) are the capacity constraints. For each failure of edge (n, m), the constraint in (10) assumes rerouted paths for arcs (n, m) and (m, n), and only trees that contain the arc failure are involved. They also consider the released bandwidth on the initial routing paths. Equations (11) and (12) are special cases of (9) for the nodes that detect the failure, node n and node m. Finally, (13) and (14) indicate that the variables take binary values.

4.2 Convexity of our model

The objective function of our model has the general form:

$$\min_{s.t.\ \ x\in C} f(x) \tag{15}$$

where C is the set Arc and f is a function over C giving the additional capacity needed for a chosen arc. The problem described by Eq. (15) is convex in the set C and the function f is convex.

Under the existence assumption of at least two paths between any pair of nodes of the graph and considering each arc of the set Arc as a segment, then the set C is convex as an intersection of convex subsets. According to [33], if the function f is affine, it is convex and the problems described by general Eq. (15) are usually stated convex problems with an implicit convexity. This implicit convexity is because there are more explicit formulations

Table 1 Restoration rate without conflicts

Networks	Number of nodes	Number of links	Number of failures	Restoration
Network1	5	7	7	7
Network2	10	18	18	14
Network3	20	31	28	23
Network4	60	81	70	61

Table 2 Restoration rate neglecting conflicts

Networks	Number of nodes	Number of links	Failures	Restoration
Network1	5	7	7	7
Network2	10	18	18	14
Network3	20	31	28	25
Network4	60	81	70	65

of convex problems such as convex optimization problems in functional form, which are convex problems of the form:

$$\min_{s.t.g_i(x)\le 0, i=1,2,\ldots m, h_j(x)=0, j=1,2,\ldots p} f(x) \tag{16}$$

Where f, $g_1,\ldots,$ g_m: $\mathbb{R}^n \to \mathbb{R}$ are convex functions and $h_1,\ldots,$ $hp.$: $\mathbb{R}^n \to \mathbb{R}$ are affine functions. Each constraint of our model can be written under any of the forms of constraints of Eq. (16). This proves that our model is implicitly convex.

5 Implementation and simulation results

Based on the comparative study in [34], the OMNet++ network simulator has many advantages: Unlike NS-2 and NS-3, OMNet++ has extensive graphical user interface (GUI) and intelligence support, provide good computation times. The flexibility of the NED language used for describing the network architecture is appropriate to meet the great topology flexibility requirements of network virtualization. OMNet++ is also able to carry out large scale network, which is an important feature for our simulations. That is why the experiments have been conducted in the simulation environment OMNet++ running on a computer with the following configuration: Core i5 2.40 GHz, 4.00 GB RAM, 12 MB cache. We applied our model to 4 networks: network1 (5 nodes and 7 links), network2 (10 nodes and 18 links), network3 (20 nodes and 31 links), and network4 (60 nodes and 81 links). These four networks satisfy the assumptions cited above. They contain a set of nodes with high degree for the estimation of the impact of multilink failures adjacent to the same node on those networks. It therefore shows the robustness of our strategy. The restoration rate of our rerouting scheme without conflict between rerouting paths is shown in Table 1. The simulations have been done on non-simultaneous multiple link failures for each tested network.

The data provided in Table 1 show that our rerouting scheme supplies rerouting solutions for almost all link failure situations considered. If the conflict constraint is neglected, we can find solutions for more failure configurations (see Table 2), but the variation is small (approximately 2%). In other words, conflict constraint does not significantly affect the number of failures handled.

Figure 7 graphically compares our rerouting scheme with that of [7]. This figure shows that the restoration rate gap between both methods increases with network size. This phenomenon is observed because the potential conflicts in a small network are not numerous, which means that cases of unsolvable conflicts are also not numerous.

We also perform an additional capacity consumption test for the previous four networks; the results are shown in Table 3. This table consists of five columns. Descriptions of the rightmost three columns are as follows: “Unused CA” represents the additional capacity available in the network; “Our used CA” represents the total of additional capacity used by our strategy for link-failure handling; and “Used by X” represents additional capacity used in the network by method [7]. The result units are expressed in seconds.

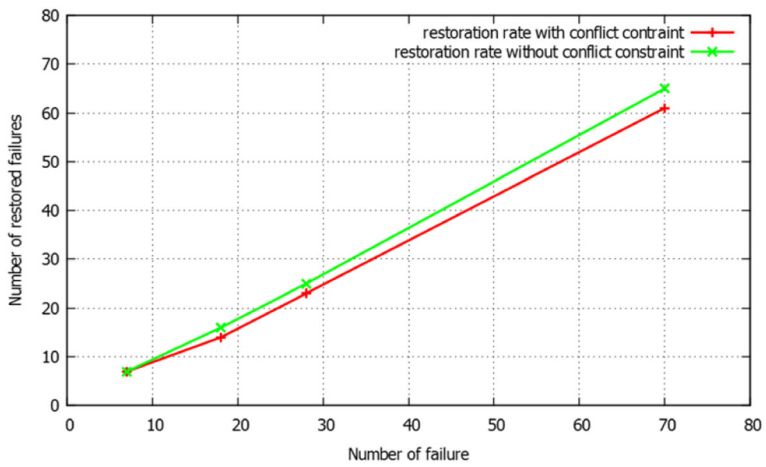

Fig. 7 Comparative graph of two versions of our rerouting scheme: with conflict constraint and without

Table 3 Comparison of our rerouting scheme with [7] concerning additional capacity used

Netwowrks	Number of nodes	Number of links	Unused CA	Our used CA	Used by X
Network1	5	7	3342	1012	1624
Network2	10	18	7216	2433	2741
Network3	20	31	14,000	3802	4524
Network4	60	81	12,052	4110	5021

The results in Table 3 show that our rerouting scheme based on a traffic-splitting strategy uses less additional capacity than does the method presented in [7]. This difference is very important when the network size increases. Figure 8 provides us a better illustration of this difference. This figure shows that the additional capacity used for flow restoration increases with network size and network connectivity.

6 Node failure problem

We speak of node failure when some flow can no longer go through a given node in the network. This situation can be caused by overflow traffic in this node or a physical failure of the given node. Because of the two-link connectivity included in our hypothesis, a node failure leads directly to the outage of at least two or several links; in other words, node failure can be treated as a simultaneous multiple link failure. In this case, failure will be detected by all nodes connected to the failed node. Figure 9 presents the failure of node number 4.

When node 4 fails, flows coming from nodes 1 and 7 must be rerouted. The failure of node 4 implies a simultaneous failure of links (1–4), (7–4) and (4–6). Therefore, we must reroute the two flows (1–4) and (7–4) to destination 6 without conflict. To solve this type of failure, two solutions are possible:

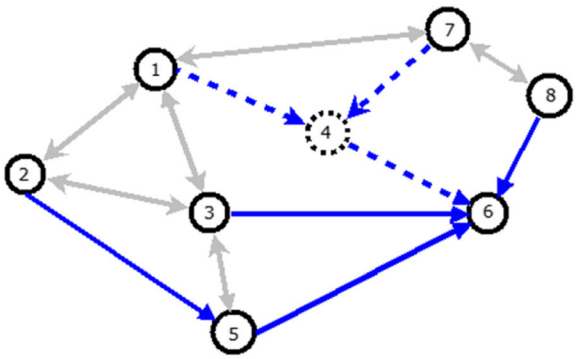

Fig. 9 Failure of node number 4

6.1 First solution resort to the controller

In the case of node failure, each switch that detects the failure sends a specific message called *packet-in message* to the controller that sets the rerouting order for the link failures related to these nodes. The idea of this rerouting approach is to solve these link failures as cascading failures. This order can be built on a node's label criteria. The nodes are labelled in a decreasing order as we approach the destination node. We could handle the failure detected by the node of a smaller label before another one with a larger label. Once the resolution order is fixed, the controller updates the routing tables of involved nodes as described in [7] by using another specific message called *packet-out message*. After this update, our rerouting scheme can be used to solve each link failure. The reaction time of this solution is too long, due to pro-activity; therefore, the principles of IPFRR are not satisfied with this solution.

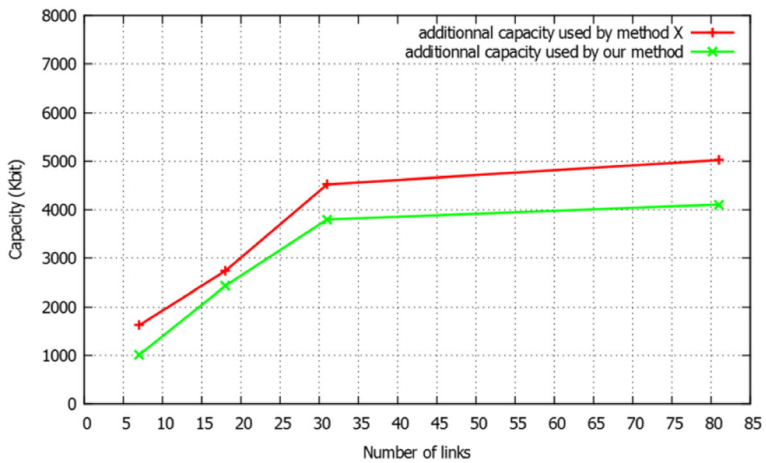

Fig. 8 Additional capacity used by different strategy

6.2 Second solution: No resort to the controller

Each link failure because of a node failure is handled locally and instantly by each node that identifies a link failure. The incurred risk in this strategy is the looping problem during flow rerouting; however, assuming our constraint imposing traffic from nodes with a lower label on another one with a higher label, cycles can be avoid. Our rerouting scheme for a simple link failure can be used to fix simple node failure situations. Recall that we speak about simple node failure when only one node fails at a time. Our rerouting strategy for simple node failure problems uses this approach. The following illustrates our rerouting scheme for a simple node failure with an example. Fig. 10 shows the nominal routing tree of an example network, and Fig. 11 presents link failures (dotted links) resulting from node 3's failure. Fig. 12 presents a cyclic problem resulting from a node failure, and Fig. 13 illustrates the efficiency of our solution to solve this cycle problem.

For each destination, we determine the nominal routing tree from each node towards this destination (see Fig. 10). The failure of node 3 generates simultaneous failures of links (5–3), (6–3) and (3–1) (dotted links). Nodes 5 and 6 will detect the breakdowns of links (5–3) and (6–3), i.e., we have two flows to reroute. These failures split the graph into two parts: the blue part and the red part (see Fig. 11).

Using our rerouting scheme, the flow coming from link (5–3) could be rerouted through arcs $(5 \rightarrow 2)$, $(5 \rightarrow 6)$ and $(5 \rightarrow 8)$. The flow of link (6–3) could follow arcs $(6 \rightarrow 8)$, $(6 \rightarrow 4)$ and $(6 \rightarrow 5)$. However, arc $(5 \rightarrow 6)$ can lead to node 3 through arc $(6 \rightarrow 3)$ or keep the rerouted flow in cycle 5–6–8-5 (see Fig. 12). Thus, arc $(6 \rightarrow 3)$ will be excluded from the list of potential paths for rerouting the flow coming from link (5–3) or node 3. If we consider the criteria related to management of the cycles, arc $(5 \rightarrow 6)$ will be considered in rerouting the paths of link (5–3), which will not be true of arc $(6 \rightarrow 5)$ (see Fig. 13).

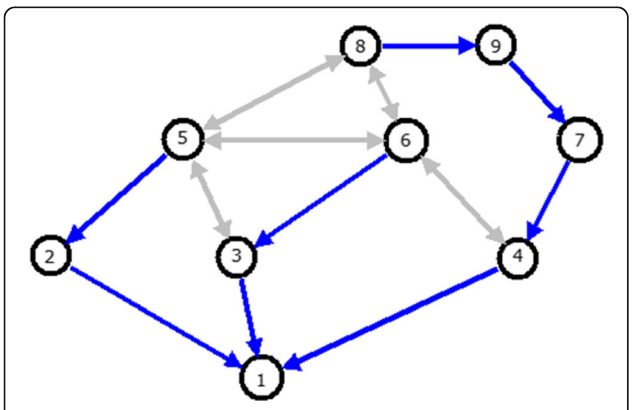

Fig. 10 Shortest path from each node to destination 1

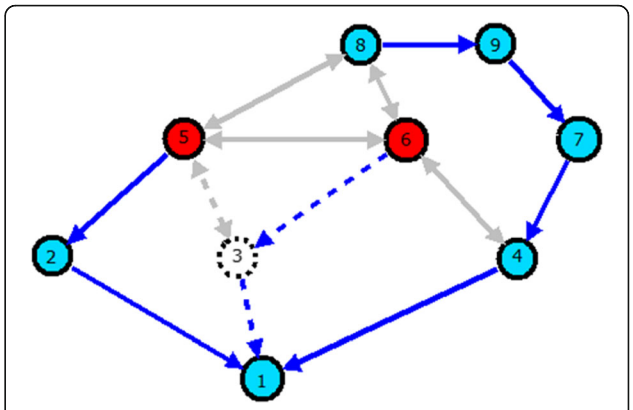

Fig. 11 Node 3's failure

Similarly, for rerouting link (6–3), arc $(5 \rightarrow 6)$ could also be used rather than $(6 \rightarrow 5)$.

Consequently, the possible rerouting paths will be 5–2-1, 5–8–9-7-4-1 and 5–6–4-1 for flow from the failure of link (5–3); concerning the flow from the failure of link (6–3), the possible rerouting paths could be 6–8–9-7-4-1 and 6–4-1. We can conclude that local reaction required by IPFRR strategy can also be preserved when addressing simple node failure situations through our rerouting scheme.

To achieve the local connectivity recovery, there is a filter similar to an agent, running inside each switch (example of OpenFlow switches) used in network architecture like ours. This agent detects the port states and acts as needed. For classical switches, there are control mechanisms provided to check that ports status.

Multiple link failures studied in the case of simple node failure involve links adjacent to that node, but we also have cases of simultaneous multiple link failures not adjacent to the same node.

7 Simultaneous multilink failures

We speak about simultaneous multiple link failures when several links fail at the same time. The case

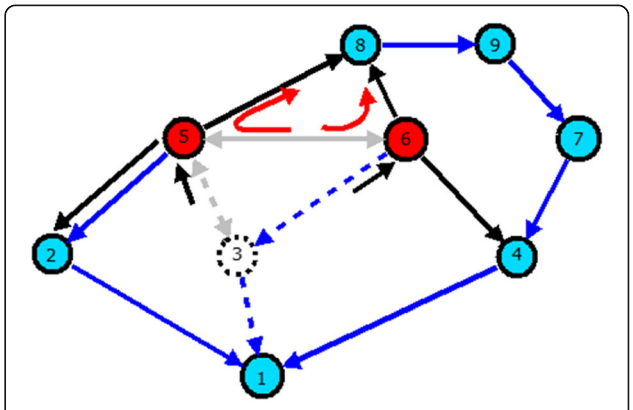

Fig. 12 Cycle example in two flows rerouting

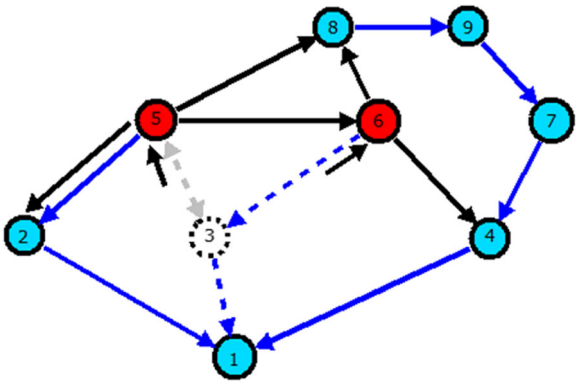

Fig. 13 Our rerouting strategy solving the cycle problem

considered in this section concerns non-adjacent links to the same single node. In this case, there are multiple nodes, each of which detects a link failure as in the simple node failure case. This type of failure can also be handled using either of two methods:

7.1 First method: Treat only one link failure at a time

In this approach, despite many link failures occurring at the same time, they are handled as non-simultaneous link failures; therefore, failures are treated sequentially. This method is used in [7], in which a rerouting scheme is proposed to solve the problem for the case of two links failing simultaneously. As stated in Section 7 about the node failure problem, the limit of this strategy is its slowness in rerouting.

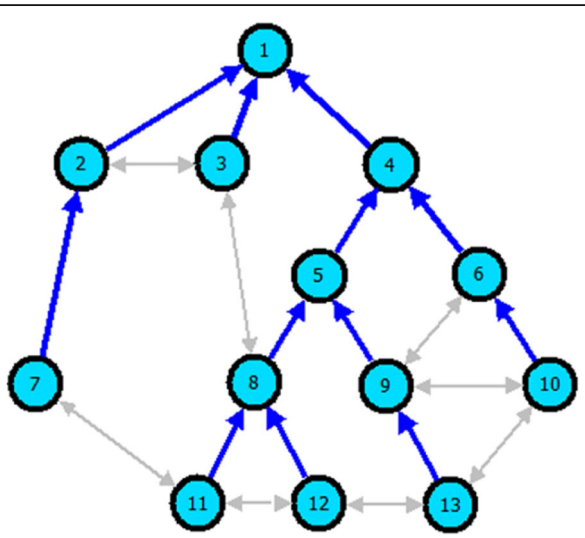

Fig. 14 An example network of simultaneous and non-adjacent two link failures

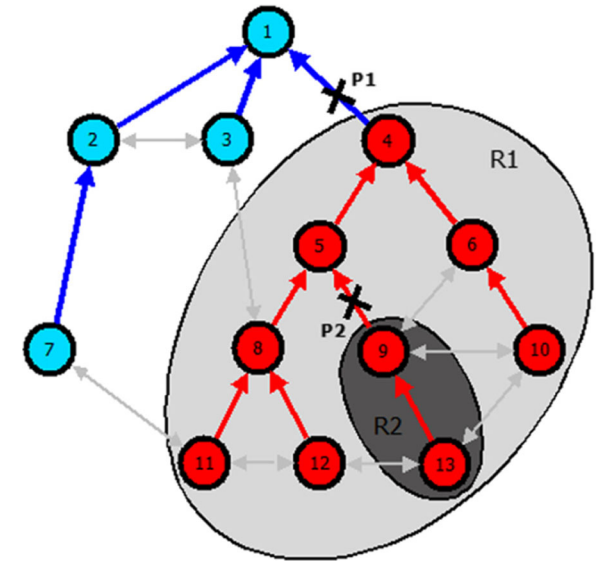

Fig. 15 Two simultaneous and non-adjacent link failure examples

7.2 Second method: Treat all link failures at the same time

This approach is similar to the second one presented in Section 7 for the node failure problem, and it enables all nodes that detect a failure to initiate the rerouting process. Our rerouting scheme for node failure can also be used here. When several link failures occur simultaneously during the rerouting process, we can use flow splitting each time to find spare capacity lacking in the network.

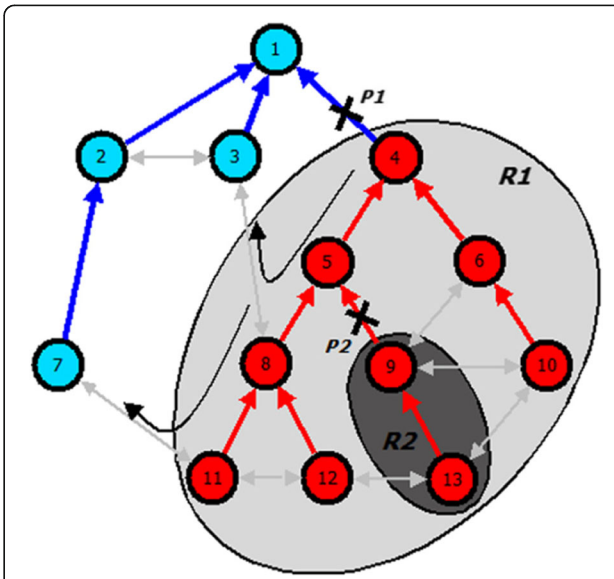

Fig. 16 Our rerouting scheme for two simultaneous and non-adjacent link failures

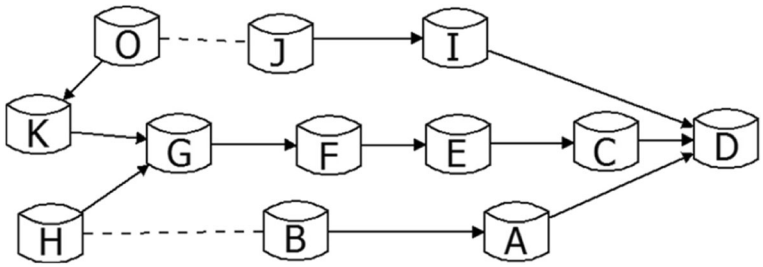

Fig. 17 Unsolvable conflict when using only one plane

Consider the example network of Fig. 14 to illustrate our rerouting strategy for the case of simultaneous and non-adjacent two-link failures.

The nominal routing tree is shown, and the destination node is labelled 1.

Figure 15 shows two link failures named $p1$ and $p2$ occurring at the same time.

The failures $p1$ and $p2$ create the red parts $R1$ and $R2$. $p1$ is detected by the node labelled 4, and $p2$ is detected by the node labelled 9. The rerouting scheme of $p1$ can be through bridges (8–3) and (11–7) leading to the paths 4–5–8-3-1 and 4–5–8-11-7-2-1 (see Fig. 16). Flows can be split at node numbers 8, 11, 2 and 3. Concerning rerouting of $p2$, link (12–13) can be considered a bridge that connects the red part R1 to R2 in addition to (8–3) and (11–7). After the link failure pair ($p1$, $p2$), if another occurs (pair ($p3$, $p4$) for example), the rerouting will be done based on the previous one to avoid conflict.

However, concerning this simultaneous multiple link failure, there are several conflict configurations that require particular attention as shown in Fig. 17. For the configuration example shown in Fig. 17, [7] affirms that the conflict problem illustrated is insoluble. Indeed, the rerouting scheme provided in [7] uses only one path for rerouting, with management of conflict between the paths similar to our strategy. We prove that this potential unsolvable conflict claimed by [7] can be solved by using multiple planes. The principle is to cross from one plane to another when there is a risk of unsolvable conflict when using a single plane.

Consider the configuration example given by [7] in Fig. 17, in which the authors claim that there is no rerouting solution. Two simultaneous link failures situations are considered: first, we have simultaneous link failures (A-D) and (C-D). Second, we have simultaneous link failures (E-C) and (I-D).

According to [7], when (A-D) fails, the only available rerouting path is A-B-H-G-F-E-C-D because if we choose path H-G-K, there would be a conflict at node G. When (C-D) fails, the traffic that comes from failure (A-D) will be rerouted by C. To reroute the traffic of failure (C-D), node C must transfer the traffic back to G; then, there are two possibilities: use G-H-B-A-D as the rerouting path, or transfer the traffic through link (G-K). We cannot use G-H-B-A because the traffic would be transferred indefinitely between A and C. Therefore, we must use F-G-K as the rerouting path in this situation.

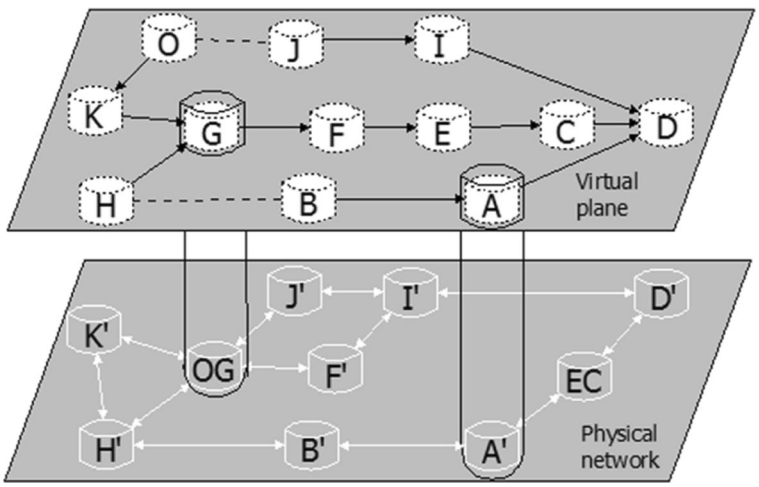

Fig. 18 Solution through multiple planes

Concerning the second situation in which the two links (E-C) and (I-D) fail at the same time, when (I-D) fails, using the same reasoning as in the previous case, the only available rerouting path is I-J-O-K-G-F-E-C-D, according to [7]. Because we used F-G-K in the previous case of link (E-C) 's failure, we must also transfer the traffic through F-G-K for this case to avoid conflict. Because both failures occur at the same time, the traffic will be transferred indeterminately between C and I; therefore, the traffic cannot be rerouted in this situation. That property is why [7] affirms that there is no rerouting scheme without conflict for destination D in this configuration.

Now, consider our rerouting scheme using multiple planes. For the same configuration example above, our rerouting solution is shown in Fig. 18. In this figure, virtual nodes E and C are hosted by physical node EC; virtual nodes O and G are lodged by physical node OG. The network topology with unsolvable conflict is located in a virtual plane.

Let us transpose the topology of Fig. 17 into the physical plane as illustrated by Fig. 18.

Assuming that the nodes which detect failures are nodes E and I in the case of simultaneous link (E-C) 's and (I-D) 's failures and considering the topology's heterogeneity in the virtual networks, node E detects that a traffic redirection through path F-G-K-O-J will be deviated on node I and cause a cyclic problem. To solve that problem, we use another plan for traffic coming from both link failures. We will choose paths E-EC-D'-D and I-I'-D'-D for link (E-C) 's and (I-D)'s failure restoration. Thus, our approach can solve unsolvable conflicts presented in [7] by making use of multiple planes.

8 Conclusion and future work

Our aim in this paper was to propose a rerouting approach to handle the single link node failure and simultaneous multiple link failure problems in a network of switches in the context of network virtualization. We proposed a conflict-free rerouting scheme that can ensure that, whatever the case of link or node failure, traffic will be rerouted to the destination. The proposed method is based on local reaction of nodes placed at the extremities of the failed link, whereas the other nodes need not know about the failure or take any particular action. Thus, the implementation is particularly easy. The flow splitting strategy used when there is insufficient spare capacity on links helps to reduce additional capacity added to the network. We proved that there exists a restoration scheme without conflict in the network and provide a mathematical model that permits calculation of the rerouting scheme with optimization of the sum of additional capacities needed for one virtual plane. We also proposed a rerouting solution using several planes to solve cases of potentially unsolvable conflicts when we use only one plane. Further work will address congestion management into the nodes implied in the rerouting and routing table updates without disturbing the network.

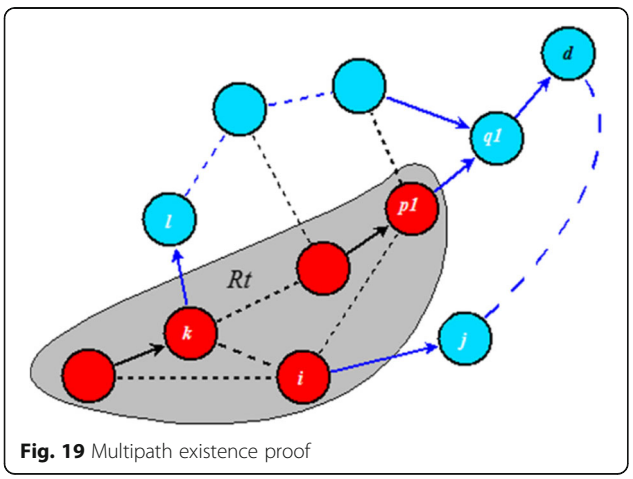

Fig. 19 Multipath existence proof

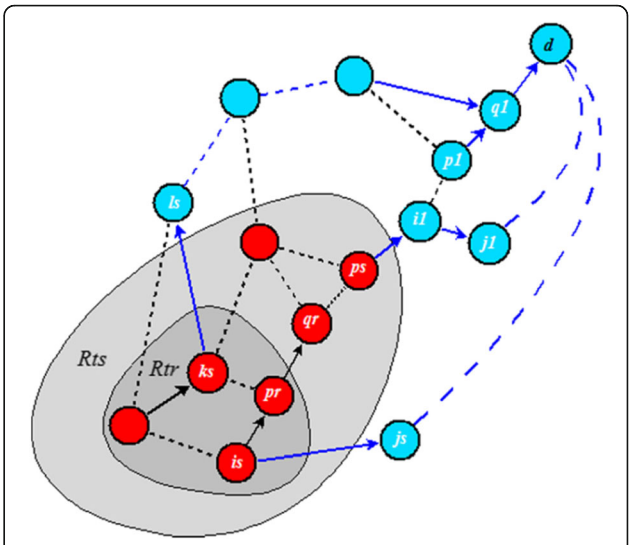

Fig. 20 Recurrence hypothesis

9 Appendix

The following is a formal proof of theorem 1. stated in Section 5. This proof is similar to the one presented in [7], with the difference that we are showing the existence of multiple paths rather than one path as in [7].

Consider a routing tree Rt to a destination d as shown in Fig. 19; then d is the sink of Rt. The dotted links represent the possible existence of nodes between nodes connected through that link. Let ($p1$, $q1$) be an arc of Rt. We assume that this arc fails and that we must find a rerouting scheme without conflict. We note that ($q1$, $p1$) does not belong to Rt, which means that, for a given

destination, the link failure problem can be treated as an arc failure. Without loss of generality in this proof, we will consider the problem of arc failure ($p1$, $q1$). $p1$ is the sink of a sub-tree $Rt1$ whose nodes are coloured in red. The other vertices in the tree are coloured in blue. We assume that all vertices are part of the tree Rt and that this assumption is true for any destination d.

Based on [7], there exists a rerouting path connecting both the red part and the blue part. Inside the red part of sub-tree ($p1$, $q1$), under the assumption of two-link connectivity, there are at least two paths going from any node of that red part to an arc connecting both the red and blue parts. Thus, if traffic splitting is performed on any of those nodes, it will be possible to reroute the flow through at least two different paths until it reaches destination d. In other words, two paths μ and ν exist from $p1$ that visit some vertices of $Rt1$ and connect a vertex of $Rt1$, which is red, to a blue vertex. This connection can be done through arcs (i, j) and (k, l) offering sufficient spare or adequate additional capacity. These arcs act as a bridge between the red and blue part. For a red vertex of $Rt1$ affected by the failure, the descended traffic will first follow paths μ and ν inside the red part to $p1$. It will then follow the original routing tree from $p1$ to i or k and use arc (i, j) or (k, l) as a bridge to reach destination d through blue vertices located in the blue part of the routing tree. Due to dimensioning issues, traffic can also be rerouted into the blue part over at least two alternative paths until destination d. According to the rerouting choices, the rerouting paths associated with the link ($p1$, $q1$) failure in the red and the blue parts are without conflict.

Consider the $n-1$ arc failures of the tree; we choose different arcs (i, j) to connect the red part to the blue part. First, we prove the existence of a bridge connecting both parts; second, we demonstrate the absence of conflict between different paths. The arcs of the tree are numbered in decreasing order as we approach sink d. We choose arcs (i, j) and (k, l) in successive order of increasing numbers. Let (pr, qr) be an arc under consideration. We assume that we have chosen arc pairs $((i_1, j_1), (k_1, l_1))$, $((i_2, j_2), (k_1, l_1))$, ... $((i_{r-1}, j_{r-1}), (k_{r-1}, l_{r-1}))$ for rerouting. pr is the root of tree Rt_r. We consider two cases. The first case is with arc (ps, qs) as the failure, and $s << r$. In this case, arcs (i_s, j_s) and (k_s, l_s) have their extremities i_s and k_s inside tree Rt_r, and their extremities j_s and l_s are out of trees Rt_s and Rt_r, which are included in Rt_s (see Fig. 16). Then, we choose arc (i_s, j_s) as arc (i, j) and (k_s, l_s) as arc (k, l) for tree Rt_r. In the second case, there is no rerouting arc with this property. We choose any arcs (i_r, j_r) and (k_r, l_r) that connect Rt_r to its complement. Each of these cases offers at least two path possibilities in the red part of the network.

Let us prove the absence of conflict in our rerouting scheme. By recurrence of the number of rerouted arcs, let us assume that we have already rerouted $n-1$ arcs in the tree. Each rerouted arc has generated at least two rerouting paths. By the recurrence hypothesis, there is no conflict for the first $n-1$ reroutings. We must verify that the n^{th} rerouting also has no conflict with the first $n-1$ reroutings.

There is no conflict by construction concerning the rerouting of the outside part of tree Rt_r, which is the part in common with the classical routing. Although this n^{th} rerouting uses the same arc as the previous ones do, in this part, the rerouting will follow the same path until destination d; therefore, there is no conflict. A conflict could occur if the splitting strategy is located in the blue part of the network. We verify no conflict exists for the part in which it goes in the opposite direction of the tree, which verifies that in the two cases cited above, there is no conflict. In the first case, when we have chosen arcs (i_s, j_s) and (k_s, l_s) in tree Rt_r, there was no conflict in that part of the tree because Rt_r would use the same arcs (i_s, j_s) and (k_s, l_s) as the passing bridges between its red part and its blue part (see Fig. 20). In the second case, the part climbing up the tree has nothing in common with the other rerouting arcs. In this case, there would exist (i_s, j_s) and (k_s, l_s). Therefore, there is no conflict in this case. We can conclude that the property remains true to the order n. The absence of conflict in our rerouting scheme can be proved by reoccurrence.

Acknowledgements
We thank the anonymous reviewers whose valuable comments and suggestions have significantly improved the presentation and the readability of this work.

Authors' contributions
JFM suggested this work. VKT and YFY carried out analysis and performed the experiments. VKT and YFY wrote the first draft of this work and worked for the revised version. JFM revised the first draft and worked on the revised version. In addition, all authors read and approved the work.

Authors' information
VKT and YFY are with the department of Mathematics and Computer Science of the University of Dschang, Cameroon. JFM is with the Computer Science Lab. MIS of the university of Picardie Jules Verne, Amiens, France.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon. [2]Computer Science Lab-Mis, University of Picardie Jules Verne, Amiens, France.

Received: 13 October 2017 Accepted: 25 April 2018
Published online: 18 June 2018

References

1. Chowdhury NM, Boutaba R. Network virtualization: state of the art and research challenges. IEEE Commun Mag. 2009;7:20–6.
2. Alkmim GP, Batista DM, da Fonseca NLS. Mapping virtual networks onto substrate networks. J Internet Serv Appl. 2013;4:3. https://doi.org/10.1186/1869-0238-4-3.
3. Bays LR, Oliveira RR, Barcellos MP, Gaspary LP, Mauro Madeira ER. Virtual network security: threats, countermeasures, and challenges. J Internet Serv Appl. 2015;6:1. https://doi.org/10.1186/s13174-014-0015-z.
4. Cheng X, Su S, Zhang Z, Shuang K, Yang F, Luo Y, Wang J. Virtual network embedding through topology awareness and optimization. Comput Netw. 2012;56:1797–813.
5. Cheng X, Su S, Zhang Z, Wang H, Yang F, Luo Y, Wang J. Virtual network embedding through topology-aware node ranking. ACM Comput Commun Rev. 2011;41:38–47.
6. Fernandes NC, Moreira MD, Moraes IM, Ferraz LH, Couto RS, Carvalho HE, Campista ME, Costa LH, Duarte OC. Virtual networks: isolation, performance, and trends. Ann Telecommun. 2011;66:339–55.
7. Pham TS, Lattmann J, Lutton JL, Valeyre L, Carlier J, Nace D. A restoration scheme for virtual networks using switches: International Workshop on Reliable Networks Design and Modeling. USA: IEEE Press; 2012. p. 800–5.
8. Pham TS. Autonomous management of quality of service in virtual networks: PhD Thesis. Compiegne: the university of Technology of Compiegne; 2004.
9. Atlas AK, Zinin A (2008) Basic specification for IP fast-reroute: loopfree alternates. https://tools.ietf.org/pdf/rfc5286.pdf. Accessed 20 July 2017.
10. Bryant S, Shand M, Previdi S (2011) IP fast reroute using not-via addresses. https://www.ietf.org/proceedings/62/slides/rtgwg-3.pdf. Accessed 10 July 2017.
11. Ho K-H, Wang N, Pavlou G, Botsiaris C. Optimizing post-failure network performance for IP fast reroute using tunnels. In: Proceedings of the 5th international ICST conference on heterogeneous networking for quality, reliability, security and robustness, article no 44. Hong Kong: ACM digital Library; 2008.
12. Kvalbein A, Hansen A, Cicic T, Gjessing S, Lysne O. Fast IP network recovery using multiple routing configurations, vol. 2006: Proceedings IEEE INFOCOM; 2006. https://doi.org/10.1109/INFOCOM.2006.227.
13. Zalesky A, LeVu H, Zukerman M. Reducing spare capacity through traffic splitting. IEEE Commun Lett. 2004;8:594–6. https://doi.org/10.1109/LCOMM.2004.833800.
14. Wang J, Nelakuditi S. IP fast reroute with failure inferencing. In: ACM proceedings of the 2007 SIGCOMM workshop on internet network management. Kyoto: ACM Digital Library; 2007. p. 268–73.
15. Kang X, Chao HJ. IP fast rerouting for single-link/node failure recovery. In: BROADNETS 2007, fourth international conference on broadband communications: Networks and Systems. USA: IEEE Press; 2007. p. 142–51.
16. Kang X, Chao HJ. IP fast reroute for double-link failure recovery. In: Proceedings of the 28th IEEE conference on global telecommunications. Piscataway: GLOBECOM; 2009. p. 1035–42.
17. Sgambelluri A, Giorgetti A, Cugini F, Paolucci F, Castoldi P. Openflow based segment protection in ethernet networks. J Opt Commun Netw. 2013;5:1066–75. https://doi.org/10.1364/JOCN.5.001066.
18. Staessens D, Colle D, Pickavet M, Demeester P. A demonstration of automafic bootstrapping of resilient openFrow networks. In Poceedings of IFIP/IEEE International Symposium on Integrated Network Managemenr (IM 2013). Ghent: IEEE Xplore Digital Library; 2013. pp. 1066–7.
19. Sharma S, Staessens D, Colle D, Pickavet M, Demeester P. OpenFlow: meeting carrier- grade recovery requirements. Comput Commun. 2013;36:656–65. https://doi.org/10.1016/j.comcom.2012.09.011.
20. Kamamura S, Shimazaki D, Hiramatsu A, Nakazato H. Autonomous IP fast rerouting with compressed backup flow entries using OpenFlow. IEICE Tans Inf Sys, pp. 2013;96:l84–192.
21. Yu M, Yi Y, Rexford J, Chiang M. Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM Comput Commun Rev. 2008;38(2):17–29.
22. Veerasamy J, Venkatesan S, Shah JC. Effect of traffic splitting on link and path restoration planning. In: The global telecommunications conference, 1994 IEEE GLOBECOM, vol. 3: Communications: The Global Bridge. USA: IEEE Press; 1994. p. 1867–71.
23. Fischer S, Kammenhuber N, Feldmann A. REPLEX: dynamic traffic engineering based on wardrop routing policies. In: Proceedings of the of the ACM CoNEXT'06. Lisboa: ACM Digital Library; 2006. p. 1–12.
24. OpenFlow multipath proposal. http://openflowswitch.org/wk/index.php/Multipath_Proposal. Accessed 26 Oct 2015.
25. Cao Z, Wang Z, Zegura E. Performance of hashing-based schemes for internet load balancing. In: Proceedings of INFOCOM'00, vol. 1. Israel: Tel Aviv; 2000. p. 332–41.
26. Prabhavat S, Nishiyama H, Ansari N, Kato N. On the performance analysis of traffic splitting on load imbalancing and packet reordering of bursty traffic. In: IEEE international Conference on Network infrastructure and digital content, IC-NIDC, USA: IEEE Press. 2009, p. 236–40
27. Bennett JC, Partridge C, Shectman N. Packet reordering is not pathological network behavior. IEEE/ACM Trans Networking. 1999;7(6):789–98.
28. Laor M, Gendel L. The effect of packet reordering in a backbone link on application throughput. IEEE Netw. 2002;16(5):28–36.
29. Leung KC, Li VO, Yang D. An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges. IEEE Trans Parallel Distrib Syst. 2007;18(4):522–35.
30. Kandula S, Katabi D, Sinha S, Berger A. Dynamic load balancing without packet reordering. ACM SIGCOMM Comput Commun Rev. 2007;37(2):51–62.
31. Adiseshu H, Parulkar G, Varghese G. A reliable and scalable striping protocol. ACM SIGCOMM Comput Commun Rev. 1996;26(4):131–41.
32. Partridge C, Milliken W. Method and apparatus for byte-by-byte multiplexing of data over parallel communications links: Patent number: 6160819, Assigned to GTE Internetworking Incorporated, December 2000, Cambridge, Massachusetts, USA.
33. Boyd S, Vandenberghe L. Convex optimization, vol. 34. UK: Cambridge university press; 2004.
34. Khana AR, Bilalb SM, Othmana M. A performance comparison of network simulators for wireless networks. USA: Cornell University, Library. 2013;arXiv: 1307.4129.