

A restoration scheme for virtual networks using switches

Thanh Son Pham
Orange Labs,
France Telecom,
Issy-les-Moulineaux, France

Heudiasyc Laboratory,
University of Technology of Compiegne,
Compiegne, France

Joël Lattmann, Jean-Luc Lutton, Laurent Valeyre
Orange Labs,
France Telecom,
Issy-les-Moulineaux, France

Jacques Carlier, Dritan Nace
Heudiasyc Laboratory,
University of Technology of Compiegne,
Compiegne, France

Abstract— Despite the success story of Internet, the ossification in the underlying infrastructure has become a key problem for its development. Network virtualization allows de-ossifying the internet and it is already a hot research topic nowadays. Virtualization is a very successful technique for sharing and reusing of resources, resulting in higher efficiency. On the other side, there is a need to cope with heterogeneity and a higher complexity in resource management. We will focus here on traffic restoration in virtual networks. The aim of this paper is to propose a simplified scalable restoration scheme for virtual networks using switches. The scheme is very easy to implement and do not need global information on the state of network. The failure is treated only locally and the other nodes in the network do not need to know nothing about it or to undertake special actions. In contrast to conventional IP routing schemes, each node routes the traffic on the basis of the entering arc and on the destination. We show that this is sufficient to deal with all link failure situations in the network, assuming that the network is two-link connected. We model the dimensioning problem with an Integer Linear Program which can be solved exactly for small networks of the literature.

Keywords—virtualization; restoration scheme; integer linear program.

I. INTRODUCTION

Nowadays Internet gains a great success in spreading its services all over the world. But more its popularity increases, more the possibility of its development decreases. It is called the phenomenon of Internet ossification. Many works have been proposed in order to solve this problem by rethinking the architecture of Internet. The approaches of network virtualization [4] prove their effectiveness in dealing with current limitations of Internet and supporting new requirements. Its principle is to implement multiple virtual routers on the same physical machine and to interconnect them through a physical network architecture. Establishing virtual networks on a physical network infrastructure comprises a superposition of different logical topologies with virtual routers. Each of the virtual networks behaves as a network in its own on which it is possible to implement different routing protocols and services.

This study is in the context of replacing all or parts of a network by switches. Networks based on switches controlled by an external controller may represent an interesting alternative to conventional router networks. Nevertheless, the absence of a direct exchange protocol between the equipments in our network could be problematic in case of failures. Our aim is to propose a rerouting strategy for network based on switches, which ensures that, for any link failure, the traffic can always be rerouted to the destination by an alternative route. Let explain in more details this strategy:

We assume that each switch is programmed with filters which permit to determine the next hop for the incoming flow. For each embedded virtual network there is a specific filter. The controller sets the flow path by programming the switches in the form of triples of type (I, N, J; where I is the source port (node), N, the current node, J, the output port (node) and F, the filter which indicates the destination. Clearly, for an incoming flow from a neighbour and a given destination, the scheme will give an output port. In case of failure, for a given destination, only one of two extremity nodes of the failed link has to react by deviating the disturbed traffic on one of its neighbours. Next, the traffic is routed according to the filter programmed in each node of the network. Hence, the proposed scheme needs only a local reaction, making its implementation particularly easy in distributed environment. This local reaction helps the network to operate normally and it can solve the problem of transient failures. We recall that a transient
failure is a failure whose duration is short, less than ten minutes, while the duration of persistent failure is longer. When the failure is determined persistent, the controller can recalculate the routing table for all nodes in the network. In order to avoid that the rerouted traffic of a failure can cause perturbations to another part of network, additional capacities are put to all arcs of the network. We introduce a mathematical model that not only can calculate the rerouting paths of network but also optimise the total sum of additional capacity. Before explaining model, we also present a proof of existence theorem, ensuring that we can find a valid rerouting scheme for a network based on switches.

The contribution of this paper is the proposal of a new scheme to solve the link failure problem in new type of networks, network of switches and the proof of existence of valid routing scheme that is resilient to link failures.

This paper is organized as follows. After this introduction, section 2 presents some related works about the link failure problem in our context. In section 3, we describe the restoration scheme and a complete example of our method. Then section 4 provides the proof of existence theorem. Our mathematical model is described in section 5. Section 6 gives some numerical results. Finally, section 7 concludes the paper.

II. RELATED WORKS

The problem of link failure is widely investigated in literature. Many works have been done for multi-protocol label switching (MPLS) [1] and real-time systems [2]. Nevertheless, taking into account requirements of virtualization environments, leads to a slightly different problematic. Our restoration scheme is pre-calculated while other protocols in network virtualization [3] are pro-active. With pre-calculated scheme, our protocol does not consume any time in rerouting the disturbed traffic in real time. In order to solve the problem of transient failure, many methods have been proposed for IP fast reroute. Nevertheless, they still have some limitations as noticed in the following:

- For loop-free alternate mechanism based methods [6], we can not be sure to be able to reroute traffic for all destinations, it only helps to reduce the number of lost packets in IP network.
- Not-via addressing [7] and tunnelling [8] mechanisms require encapsulation and decapsulation of packets, while in multiple routing configurations mechanism [9], the packets need to carry configuration information. With the appearance of optic network, these methods which modify the packets are not recommended.

Although [5] does not have the limitations cited above, it uses the condition of our network as its principle. Indeed, this method can determine the next hop of traffic when destination and entering interface of traffic are provided. [5] calculates the rerouting path by using the metrics and the shortest paths while our method constructs the path in optimising the additional capacity and the filter is not necessarily using the shortest path. The current work considers only one virtual network. The generalization of the problem will be done in future.

III. A NEW LINK FAILURE RESTORATION SCHEME

In this paragraph, we will present our scheme of routing and rerouting in case of failure. In case of traffic for a given destination, the routing is associated with a tree to that destination, the tree is constructed by a given criteria, for instance it can be the shortest path tree. We suppose that the routing is given. In case of failure of an arc or edge (both arcs are then concerned), we reroute the traffic through an alternative path. According to the routing scheme, for two independent failures, if two rerouting paths to a given destination have an arc in common, they have to merge after this arc. This requirement holds for both nominal and rerouting paths. If two paths do not satisfy this requirement, they are in conflict. Any routing scheme satisfying this requirement is said without conflict. We recall that in case of failure, only the extremity failed link nodes will know about the failure and initiate the traffic diversion while all other nodes in the network will apply the filter for each incoming flow without making any difference between disturbed and non disturbed flows. Because the disturbed traffic is rerouted on a single alternative path while they should satisfy the conflict requirement, the cost, in term of resources and of computational times, is expected to be higher compared to conventional schemes using multipath rerouting.
We present in this section a complete example of our restoration scheme. We can see in Figure 1 the original graph. If we apply to all nodes of the graph a filter with respect to a destination D, the traffic flow that corresponds to this filter uses a directed tree which converges to destination D (Figure 2). When the link (S, D) fails, the routing tree will be divided into two parts: the Red part, whose nodes are filled, and the Blue part, whose nodes are unfilled (Figure 3). All the traffic to destination D that passes by node S cannot be transferred by link (S, D), it will be rerouted by the path (S, T, U, V, K, D) pre-calculated in our restoration scheme (Figure 4). This alternative path connects the Red part to the Blue part and it will not interfere with the nominal routing. In this scheme, only node S knows that there is a failure at link (S, D), other nodes operate normally as programmed. We can see that the traffic to destination D passing by arc (T, S) is rerouted through arc (S, T) but this will not cause any looping problem because of the configuration of filters. When the node T receives traffic coming from U and knows that its destination is D, T will transfer the traffic to node S. In case of link failure, node S will transfer all traffic to T. Knowing traffic coming from S and its destination D, T will transfer all traffic to U, which in its turn will transfer it to V. The traffic will be transferred from V to K then to destination D. Therefore, there is no looping problem and the traffic is rerouted without causing any conflict.

IV. THEOREM OF EXISTENCE OF A RESTORATION SCHEME

In this section we study the question of existence of a solution of rerouting without conflict. Let start by specifying the assumptions. The graph is assumed to be oriented and symmetric with capacities on the arcs. There are at least two disjoint-arc paths between any two nodes of the graph. There is only one link failure at a time.

Theorem 1: There exists for any destination l a rerouting plan without conflict.

Proof: Let A be the routing tree to the destination l, then l is the sink of A. Let (p1, q1) be an arc of A. We assume that this arc fails and we must find a rerouting scheme without conflict. We can remark that (q1, p1) does not belong to A, which means that for a given destination, the problem of link failure is the same as arc failure. Without loss of generality, we will consider in this proof the problem of arc failure (p1, q1). p1 is then the sink of a sub-tree A1 whose nodes are colored in red. The other vertices in the tree are colored in blue. Without loss of generality we assume that all vertices are part of the tree A and this is true for any destination l.

We know that there are at least two disjoint paths in the initial graph going from p1 to l. Under the assumption of two-link connectivity, all the top fraction containing p1, but without the vertex l contains at least two outgoing arcs. Therefore there are at least two arcs outgoing of A1 (Figure 5). Since one of these arcs is (p1, q1), there exists a path µ from p1, that visits the vertices of A1, and connects a vertex of A1, which is red, to a blue vertex. So there is at least one arc (i, j) of µ connecting the red vertices to the blue vertices (Figure 5). We associate with this arc a rerouting path for the failed arc. Let k be a red vertex of A1 which is affected by the failure. Traffic to destination l and coming from k goes first to p1, then it follows the original routing tree from p1 to i. It uses arc (i, j), then from j to destination l follows the original routing tree. According to the choice of rerouting, the rerouting of various paths associated with the breakdown of (p1, q1) are without conflict.

Now we will consider the n-l failures of arcs of the tree and choose the different arcs (i, j). We number the arcs of the tree so that their numbers are decreasing as we approach the sink l. We choose consequently the arcs (i, j) in successive order of increasing numbers. Let then (p1, q1) an arc under consideration. Assume that we have chosen arcs (i1, j1), (i2, j2) ... (in-l, jn-l) for rerouting. p1 is the root of tree A1. We consider two cases. The first case is when we have chosen for an arc (p1, q1) with s strictly smaller than r an arc (i, j) whose extremity i is in the tree A1. In this case the other extremity j is out of the tree A1. and a fortiori outside of the tree A, which is included in A (Figure 6). We will choose then arc (i, j) as arc (i, j) for the tree A. Note that there exists at most one rerouting arc with this property (proof by contradiction). In the second case, there is no rerouting arc with this property. We choose any arc (i, j) that connects A1 to its complement.

We must show that the rerouting has no conflict. We demonstrate it by recurrence on the number of rerouted arcs. We therefore consider that we have already rerouted r-l arcs in the tree. By recurrence hypothesis, there is no conflict for the
first \( r-1 \) reroutings. We verify that the \( r \)-th rerouting has also no conflict with the first \( r-1 \) reroutings. There is no conflict by construction regarding the rerouting of the outside part of the tree \( A_r \), that is to say the part in common with the classical routing. Even if it uses the same arc, in this part it will follow the same rerouting path till destination \( l \), so it is without conflict. Also we verify that there is no conflict for the part where it goes in the opposite direction of the tree, which means to verify that in the two cases cited above there is no conflict. In the first case when we have chosen an arc \((i_o, j_o)\) in the tree \( A_r \), there is not any conflict in that part of the tree because \( A_r \) will use the same arc \((i_o, j_o)\) as a passing bridge between its red part and its blue part. In the second case, the part of climbing up the tree has nothing in common with the other arcs of rerouting. Otherwise there would exist \((i_o, j_o)\). Therefore, there is no conflict in this case neither. We can conclude that the property remains true to the order \( r \). This demonstrates by recurrence the absence of conflict.

V. MATHEMATICAL MODEL

We use the assumptions given in the previous section. Below we are giving the notation used in our model:

- \( A^l \): set of arcs of the routing tree to the destination \( l \).
- \( r_{ab} \): additional capacity assigned to the arc \((a, b)\).
- \( T^l_{ij} \): total traffic for \( l \) that passes through the node \( v \), \( v \) is the node that detects the failure. In fact, the failure is characterized by a source \( v \) and a destination \( l \), because we use the routing tree for nominal routing. For a destination \( l \), the failed arc is the one routing the traffic going to \( l \) and coming from \( v \) by the nominal routing.
- \( 0 \): indicates a fictive node used to divert traffic in case of failure. We introduce the fictive node \( 0 \) that will be used for all failures. For a given failure \((v, l)\), the traffic to \( l \) will be rerouted by a single path from \( 0 \) to \( l \) and starting with the arc \((0, v)\).
- \( Red^l \): sub-tree of sink \( v \). Recall that in case of failure, the tree is divided into two parts, the isolated part, that is the Red part, and the Blue part. The alternative path will reroute traffic from Red part to Blue part.
- \( Blue^l \): \( A^l - Red^l \).
- \( y_{ijk}^v \): this binary variable indicates whether the alternative path to destination \( l \) and for a given failure contains arcs \((i, k)\) and \((k, j)\), the node \( v \) is the node that detects the failure.
- \( x_{ijk}^v \): this binary variable indicates the rerouting scheme to destination \( l \). The one takes value 1 if there exists a failure whose alternative path to destination \( l \) contains arcs \((i, k)\) and \((k, j)\). It means this variable takes value 1 if there exists \( v \) that \( y_{ijk}^v \) is equal to 1.
- \( a_{ab}^{uv} \): binary coefficient equals 1 if the arc \((a, b)\) belongs to the path in nominal routing from \( v \) to \( l \) except the failed arc.
- \( \text{Triple} \): All triples \((i, k, j)\) where \( i, k, j \) are nodes of the graph, \( i \) can be the fictive node, and \((i, k)\) and \((k, j)\) are two adjacent arcs, \( i \) is different from \( j \).

\[
\min \sum_{(a,b) \in Arc} r_{ab} \quad (1)
\]
\[
\sum_{v \neq 0, v \neq l, v \in V} y_{ijk}^{vl} = 1, \ v \in V, \ l \in V \quad (2)
\]
\[
\sum_{j \in \text{parent of} \ k} x_{ijk}^l \leq 1, \ (i, k) \in Arc, \ l \in V \quad (3)
\]
\[
y_{ijk}^{vl} = 0, \ l \in V, \ v \in V, \ i \in Red^v, \ (i, k) \in A^l, \ (i, k, j) \in \text{Triple} \quad (4)
\]
\[
\sum_{i \in V \setminus \{i, k, j\} \in \text{Triple}} y_{ijk}^{vl} \leq 1, \ l \in V, \ v \in V, \ (k, j) \in Arc \quad (5)
\]
\[
\sum_{i \in \text{parent of} \ v} y_{ijk}^{vl} = \sum_{j \in \text{parent of} \ k} y_{ijk}^{vl}, \ l \in V, \ (i, k) \in Arc, i \neq 0, i \neq v, k \in V \setminus \{l\} \quad (6)
\]
\[
y_{ijk}^{vl} = y_{ijk}^{vl} + y_{ijk}^{vl} - y_{ijk}^{vl}, \ l \in V, \ v \in V, \ (v, k) \in Arc \quad (7)
\]
\[
y_{ijk}^{vl}, y_{ijk}^{vl} - y_{ijk}^{vl} \geq 0, \ j \in Blue^v \setminus \{l\}, (j, j) \in A^l, (k, j) \in A^l, v \in V, \forall (i, k, j) \in \text{Triple} \quad (8)
\]
\[
\sum_{v \in V} y_{ijk}^{vl} \geq x_{ijk}^l \geq \frac{\sum_{v \in V} y_{ijk}^{vl}}{\text{cardinal}(V)}, \ (i, k, j) \in \text{Triple}, \ l \in V \quad (9)
\]
\[
\sum_{l \in V \setminus \{v, w\} \in A^l} \sum_{i \in \text{parent of} \ v} \sum_{j \in \text{parent of} \ k} y_{ijk}^{vl} T^l_{ij} \leq \sum_{l \in V \setminus \{v, w\} \in A^l} \sum_{i \in \text{parent of} \ v} \sum_{j \in \text{parent of} \ k} y_{ijk}^{vl} T^l_{ij} \quad (10)
\]
\[
y_{ijk}^{vl} T^l_{ij} \leq y_{ijk}^{vl} + \sum_{l \in V \setminus \{v, w\} \in A^l} \sum_{i \in \text{parent of} \ v} \sum_{j \in \text{parent of} \ k} a_{ijk}^{vw} T^l_{ij} \quad (11)
\]
\[
y_{ijk}^{vl} T^l_{ij} \leq y_{ijk}^{vl} + \sum_{l \in V \setminus \{v, w\} \in A^l} \sum_{i \in \text{parent of} \ v} \sum_{j \in \text{parent of} \ k} a_{ijk}^{vw} T^l_{ij} \quad (12)
\]
\[
x_{ijk}^l \in \{0, 1\}, \forall (i, k, j) \in \text{Triple}, \forall l \in V \quad (13)
\]
\[
y_{ijk}^{vl} \in \{0, 1\}, \forall (i, k, j) \in \text{Triple}, \forall l \in V, \forall v \in V \quad (14)
\]

(1) gives the objective function. Recall that the objective is to minimize the sum of additional capacity allocated to each arc. This objective function will allow us to evaluate the ratio between the additional capacity and the installed capacity.

The constraint (2) implies that there exists exactly one arc going out from \( v \) for each disturbed traffic from \( v \) to \( l \). Because
the path is unique, the flow on arc \((0, v)\) must be \(1\) (Figure 7), and \((v, j)\) must not be the failed arc.

The constraint (3) ensures that there is no conflict in terms of rerouting, that is to say there is only one exit for the incoming traffic. If we take the arc \((i, k)\), to route to the destination \(l\), there is at most one output \((k, j)\) (Figure 7).

To avoid the problem of looping and conflict, the alternative path should not contain any arc of nominal routing in the Red part of network. (4) assures that condition.

Constraint (5) assures that there will be no looping in the network. Indeed, for a given destination and a given failure, the alternative path contains a loop when there are two entering arcs and one outgoing arc. This constraint prohibits this kind of problem.

Constraints (6), (7) and (8) are the flow constraints that ensure the continuity of the alternative path. (6) is the constraint of flow conservation. As it can be seen on Figure 7, the total amount of traffic coming into \(i\) equals the total outgoing traffic of \(k\), because there exists only one alternative path for a failure and a destination, there is only one incoming stream and one outgoing stream. This type of flow conservation does not hold for the node \(v\) that detects the failure; we therefore have a constraint of flow (7) for this case. In the blue part, if the path uses an arc of the initial routing, it must continue to destination \(l\), we thus have (8).

(9) implies the relationship between two rerouting paths that avoids a conflict (see definition of variables \(x\) and \(y\)). Because \(x\) and \(y\) are binary variables, with the same triple \((i, j, k)\) and same destination \(l\), 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.

(10), (11), (12) are the capacity constraints. For each failure of edge \((v, w)\) (Figure 7), constraints (10) consider rerouted paths for both arcs \((v, w)\) and \((w, v)\), and only trees that contain the arc failure are involved. They also take into account the released bandwidth on the initial routing paths. (11) and (12) are special cases of (9) for the nodes that detect the failure \(v\) and \(w\). Finally, (13) and (14) indicate that the variables take binary values.

VI. NUMERICAL RESULTS

The numerical results we have obtained are computed with IBM ILOG OPL IDEA which uses IBM ILOG CPLEX 12.1.0.

The calculations are executed with a virtual computer with the following configuration: Quad Core 1.8GHz, 8.00 Go RAM, 12Mb cache.

We have applied our model to 4 networks, a test network (7 nodes, 9 links) and 3 real networks: Atlanta (15 nodes, 22 links), Polska (12 nodes, 15 links), Nobel-Germany (17 nodes, 26 links). Theses 4 networks satisfy the assumptions cited above.

The table I has 5 columns: the name of network, the number of nodes and the number of links in the network, the number of constraints and the number of variables in the model. We can see that the number of constraints and variables is very large when the size of network becomes larger or the network is highly meshed.

The table II has 3 columns: the name of network, the ratio capacity and the calculation time (hour: minute: second). The
ratio capacity column describes the ratio between the added capacity and the installed routing capacity.

<table>
<thead>
<tr>
<th>Networks</th>
<th>Nodes</th>
<th>Links</th>
<th>Constraints</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>7</td>
<td>9</td>
<td>1951</td>
<td>2419</td>
</tr>
<tr>
<td>Polska</td>
<td>12</td>
<td>18</td>
<td>10976</td>
<td>16567</td>
</tr>
<tr>
<td>Atlanta</td>
<td>15</td>
<td>22</td>
<td>20371</td>
<td>30781</td>
</tr>
<tr>
<td>Nobel-German</td>
<td>17</td>
<td>26</td>
<td>57587</td>
<td>49828</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Networks</th>
<th>Ratio Capacity</th>
<th>Calculation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>0.96</td>
<td>0:0:1</td>
</tr>
<tr>
<td>Polska</td>
<td>0.85</td>
<td>0:22:10</td>
</tr>
<tr>
<td>Atlanta</td>
<td>1.08</td>
<td>0:1:7</td>
</tr>
<tr>
<td>Nobel-German</td>
<td>1.16</td>
<td>5:12:24</td>
</tr>
</tbody>
</table>

VII. CONCLUSION

We have presented a rerouting approach that can handle single link failures in a network of switches. The proposed method is based on local reaction of nodes placed at the extremities of the failed link, while the other nodes do not need to know or take any particular action. This makes the implementation particularly easy. We have proved that there exists a restoration scheme without conflict in the network. We have also provided a mathematical model that permits to calculate the rerouting scheme and optimise the sum of additional capacities. Our work is only for one plan of virtual network. Further work is needed for extending the method to several virtual networks sharing network physical facilities.

REFERENCES