High Quality and Resilient Backbone IPTV Networks

André Ricardo Simões Bento

Abstract—Multicasting of TV signals with the IP protocol suite is a challenge. IP was initially designed for unicast not multicast. The routing and transport protocols of IP were not planned for applications requiring a combination of steady high bandwidth, low packet delay and low packet loss.

This thesis focuses on the design of broadcast trees for the backbone of IPTV networks. We will present algorithms for the calculation of optimal out-branchings and resilient out-branchings, where in case of link failure, it is possible to substitute the link by a backup path, granting that the other links of the tree are not affected by the failure. The failure recovering is provided by Fast Reroute mechanisms, which are capable of handling failure in sub-second time.

Index Terms—IPTV, Backbone, Minimum Out-Branchings, Resilient Out-Branchings, Fast Reroute, MPLS

I. INTRODUCTION

A. Problem Statement

The objective of this thesis is to compute and implement different out-branching (out-branching is the term we use to define a spanning directed tree) designs, understanding the advantages and disadvantages of these implementations in the multicast of IPTV streams.

Our goal is to design out-branchings that can improve the end user’s Quality of Service (QoS) always having in mind the requirements of distributing IPTV packet streams, namely:

• Minimal Packet Loss: IP packet loss can represent various levels of image damage, ranging from a single pixelated block in an image, to a long period of degraded, pixelated or unavailable sequence of images. In order to offer the user a satisfactory Quality of Service (QoS) packet loss needs to be minimal.

• Steady High Bandwidth: The total amount of video-stream data that can be sent is ultimately limited by the bandwidth provisioned over the network. Any increase in bandwidth demand that is above the maximum capacity of the link, will result in video packets being lost.

• Low Packet Delay: Live TV streams are encoded and sent in IP packets. At the reception, an initial delay is introduced and the first packets are stored in a buffer. After the initial delay, packets are decoded and transmitted in the user’s TV. This mechanism helps in coping with short transmission delays and jitter. However, if a packet fails to arrive at the reception at the time it is supposed to be decoded, it is like it was lost since it will not be used.

• Sub-second Failure Restoration: In the case of link failure, restoration needs to be done within few milliseconds to avoid packet loss, which could cause image damage or even service interruption.

B. Contributions

The objective of this thesis is to compute and implement different out-branching designs, understanding the advantages and disadvantages of these implementations. Given a certain out-branching we purpose a protocol suite that can use this out-branching to transport IPTV packets and, at the same time, is capable of handling the requirements of IPTV. We have studied the interaction between the involved protocols and based on previous work ([1], [2], [3] and [4]), developed a strategy to operate them.

We have also studied different out-branching designs:

• Shortest path out-branchings, where the cost of the paths from the root to the each vertex is minimum;

• Minimum cost out-branchings, where the sum of the costs of the links in the out-branching is minimum;

• Resilient out-branchings, where each link of the out-branching can be protected by a path that is disjoint from the out-branching. If the protection paths shared links with the main tree, these links would transport more than packet stream replicas, this could cause link overload.

We have implemented algorithms that generate the out-branchings above. Regarding the resilient out-branchings we will present 2 algorithms. Given a certain graph, one of them generates 2 disjoint out-branchings and the other generates the maximum number of link-disjoint out-branchings possible. For the first, we have come up with 2 heuristics that improve the resultant out-branchings. The purpose of the heuristics is the following:

• Improve the main out-branching: where the objective is to reduce the link cost of the paths between the root and the vertices in the main out-branching;

• Improve for the backup paths: where the objective is to reduce the total link cost of paths used for backup.

As far as the protection paths are concerned, we also purpose a way of calculating them.
C. State of the Art

A group of authors ([1], [2], [3] and [4]), suggested a protocol suite capable of accommodate custom out-branching designs, as well as a failure handling method used to avoid packet loss during the Fast Reroute recovering process. The authors have also suggested an algorithm to calculate out-branchings, and respective disjoint protection paths. This algorithm took care of link overload problem in case failure, however, the authors were not preoccupied with the cost of the out-branchings. They also did not had in consideration the cost of the protection paths. Other authors came up with algorithms [5] that resulted in similar out-trees, also disregarding the cost optimization aspect.

In [6] and [7] simple schemes are proposed to solve single link failures in sub-second time with low packet loss in the process, these schemes introduce minor changes in routing and are easy to implement. However, these schemes disregard the link overload problem during the recovery. Like the others the proposed solutions do not account for the costs of the out-branchings.

Another different solution is proposed in [8], this scheme uses fast layer 3 failure detection, this type of schemes is often disregarded by the telecommunications carriers because they require short timers to detect the failures, this can produce false positive cases and as we will see further ahead can compromise the best routing options.

Finally in [9], 3 protocol architectures are discussed with the objective of distributing IPTV streams, the advantages and disadvantages of the architectures are briefly explained.

II. Protocol Architecture

A. Setting up a pre-calculated out-branching

Given a certain out-branching we purpose a protocol suite that can use this out-branching to multicast IPTV packets and, at the same time, is capable of handling the requirements of IPTV. This protocol scheme uses PIM-SSM (Protocol Independent Multicast Source Specific Mode) combined with P2MP LSPs (Point-to-Multipoint Label Switch Paths).

PIM-SSM is used to coordinate the multicast groups and generate the multicast out-branchings. IGP (Interior Gateway Protocol) routing is assured by either IS-IS (Intermediate Systems Intermediate Systems) or OSPF (Open Shortest Path First). P2MP LSPs are established using RSVP-TE to enable resource reservation and FRR (Fast Reroute) is used for rapid link restoration using local protection. Routes are established following the next steps:

1. The network topology and its current link costs are fed to an out-branching calculation algorithm that will be in the next chapter. The routers can run the programmed algorithm the complete network topology. The out-branching calculation can also be done by the network operator;
2. After finding the out-branching, we need to re-set the IGP link costs. The cost of the physical links in the main out-branching remains the same. The costs of the physical links that are not in the main out-branching are set to $\infty$. The new IGP costs are then downloaded to the routers;
3. This way PIM-SSM will generate the same out-branchings that we have calculated;
4. RSVP signals the out-branchings generated by PIM-SSM, making the proper resource reservations, forming a P2MP LSP.
5. The backup paths for each link of out-branching, can be calculated by the network operator and downloaded to the routers. Alternatively, RSVP can setup them automatically using the IGP routing information. To do such, IGP costs need to be re-set again. The cost of the physical links of the main out-branching are set to $\infty$ and the costs of the other physical links are set to its original value. Doing such grants that the backup paths that are calculated do not share any link with main out-branching.
6. If the backup paths were calculated using RSVP, the IGP costs need to be set to the values of step 2;

B. Failure recovering process

We have described a procedure to assure the setup of the pre-calculated out-branchings and protection paths. Fig. 2 shows a network that has already been set.

Fig. 2 – Setup of calculated out-branchings

The normal black lines represent physical links, which provide connectivity in both directions. The red arrows represent the links used by the out-branching. The dashed
black lines represent FRR backup paths and the bold red arrows represent the links that are protected. For simplicity only two backup paths are presented path R1-R3-R2 is protecting link R1R2 and path R4-R5-R7-S-R1 is protecting link R4R1. Note that the protection paths are disjoint from the main tree. They are built this way to prevent congestion in main tree links. If the protection paths shared links with the main tree, these links would transport more than packet stream replicas. For example, if we used path R4-R3-R1 to protect link R4R1, link R4R3 would transport the packets destined to R3 as well as the packets destined to R1.

Let us assume that link R1R2 fails. Once the failure is detected the respective FRR backup is activated. Fig. 3 shows the label switching table of node R1 and the activation of the backup path. Before the failure packets arrived at R1 through interface C with label 50, R1 switched the packet label to label 40 and would send it to R2 through interface A. When the failure is detected, R1 uses the FRR columns of the switching table. It sends the packet using interface B with label 30.

The activated protection path is transparent to the IGP protocol, which is not yet aware of the link failure. The process of detecting the failure and activating the protection path takes about 50 ms, which is lower than the failure detection times of the IGP and PIM protocols. Both OSPF and IS-IS use an HELLO based protocol that take about 1-10s to detect link failure.

Normally when the failure is detected, LSAs are broadcasted to inform the routers. After the broadcast, the IGP routing table of S would be like the one represented below.

<table>
<thead>
<tr>
<th>Router</th>
<th>Path Cost</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>20</td>
<td>B</td>
</tr>
<tr>
<td>R2</td>
<td>X</td>
<td>B</td>
</tr>
<tr>
<td>R3</td>
<td>21</td>
<td>B</td>
</tr>
<tr>
<td>R4</td>
<td>18</td>
<td>B</td>
</tr>
<tr>
<td>R5</td>
<td>8</td>
<td>B</td>
</tr>
<tr>
<td>R6</td>
<td>14</td>
<td>B</td>
</tr>
<tr>
<td>R7</td>
<td>3</td>
<td>B</td>
</tr>
</tbody>
</table>

Table 1 – Router S IGP routing table

S is the head-end of the sub-LSP that S-R7-R6-R4-R1-R2, which was signalled with RSVP and needs to be refreshed periodically. When RSVP tries to refresh the LSP it uses the IGP routing information. This would generate an error since S is not aware that the backup path is being used. This error would cause the LSP to be teardown.

The purpose of the backup is to protect traffic while the LSP head end looks for an alternate path for the LSP, avoiding the failed link. For this to happen, the head end must first find out about the failure. The PLR takes care of this by notifying the head using an RSVP Path Error message with a ‘Notify’ error code and ‘Tunnel locally repaired’ subcode. In addition to this, a new flag indicating that the path is locally repaired is turned on in the Record Route Object. If an LSA is received by the head end the RSVP Path Error is received, the LSA must be suppressed.

After the broadcast of the link failure, IGP costs are recalculated and a new topology is found. During this process the backup path is at use. The new topology and link costs need to be fed to the out-branching calculation algorithm again, so that a new out-branching can be calculated.

The IGP costs are then downloaded to the routers that use PIM to generate the out-branchings and RSVP to signal them. After the establishment of the new LSPs, the old are torn down.

### III. DESIGNING OUT-BRANCHINGS

We present 3 different algorithms with different objectives, which given a specific network, calculate different types of out-branchings.

#### A. Algorithm 1: Finding minimum cost out-branchings (MOBs) in a multigraph

Given a certain graph $G$ with attributed link costs, the MOB of $G$ is the out-branching that has the lowest cost, among the existent out-branchings in $G$. The objective of algorithm 1 is to find the MOB of a given multigraph.

This algorithm makes use of a property of 2-connected symmetric graphs.

Proposition: Let $G$ be a 2-connected symmetric graph and $P$ an arbitrary simple path, between vertices $s$ and $t$. We can remove $P$ from $G$ that there will still be a path left between $s$ and $t$.

Proof: Let us consider two disjoint sets, $Z \in V(G)$ and $Y \in V(G)$. In a 2-connected graph, there are 2 pairs of symmetric links. Let us suppose by absurd that we can find a simple path between two arbitrary vertices, $s \in Z$ and $t \in Y$, given that such path uses all links of $E^-(Y)$. Where $E^-(Y)$ represents the set of entering links of $Y$.

Fig. 4 depicts sets $Z$ and $Y$, each having two entering links. In this case path $P$ uses all the links of $E^-(Y)$, but $P$ is not a simple path.
In fact, in order to use links $e$ and $g$ in the same path, we must use either link $f$, or link $h$. Given so, the resultant path will not be a simple path.

Algorithm 1 recursively changes the graph until every vertex has an entering link that has cost equal to 0 and there are no cycles in the graph with total link cost equal to 0. At this point, the algorithm recursively selects links to be part of the MOB and returns the graph to its original form.

The first step is to reattribute the costs of the links. For each vertex $u$ we find the cost of the minimum cost link $y_u$, among the links that are entering vertex $u$. After that we need to reattribute the costs of the links entering $u$, by subtracting its cost with $y_u$.

$$c'(vu) = c(vu) - y_u$$

where $c'(vu)$ represents the reattributed costs and $c(vu)$ represents the original cost of link $uv$. After this step, for every out-branching $T$ rooted at root $s$ in graph $G$ the sum of the reattributed costs added to the sum of costs of the minimum links entering each vertex, will be equal to the sum of the original link costs.

$$\sum_{u \in S} y_u = \sum_{u \in E(T)} c(vu) - \sum_{u \in E(T)} c'(vu)$$

Thus, $T$ is a MOB in relation to costs $c$ if, and only if, it is a MOB in relation to costs $c'$. This step is depicted in Fig. 5.

The second step of algorithm is to find cycles with links that have cost equal to 0. After that the algorithm agglomerates them into a single super node, as represented in Fig. 6.

We then need to repeat the first two steps until the following two conditions apply:

- There are no cycle with 0 cost links left;
- Every vertex has an entering link that has cost equal to 0.

After this we will end up with graph on the right of Fig. 7.
Next we will start to the links of the out-branching. The first that are going to be selected are the ones that have cost equal to 0 (bold red links Fig. 8).

![Fig. 8 - Algorithm to find MOBs: Step IV (0 cost links are selected as part of the MOB)](image)

Finally we need to expand the supernodes and select a special link \( f \) that will not be included in the out-branching. Let us define \( e \) as the entering link of a supernode and \( u \) as the vertex to where link \( e \) points to. The special link \( f \) is link contained in the cycle that also points to vertex \( u \).

![Fig. 9 - Algorithm to find MOBs: Step V (the links of the cycle except link \( f \) are added to the MOB)](image)

The complexity of this algorithm is \( O(|V(G)|(|E(G)| + |V(G)|)) \).

**B. Algorithm 2: Finding 2 link-disjoint out-branchings in 2-connected symmetric graphs**

This algorithm generates two disjoint out-branchings, so that, in case of failure, it is possible to use an alternative out-branching to distribute the packets.

Let \( G \) be a 2-connected symmetric graph, where a link has always its symmetric: another link in the reverse direction. Let \( s \) a source vertex, \( R \subseteq G \) and \( B \subseteq G \) the two resultant disjoint out-branchings. This algorithm finds two link-disjoint out-branchings rooted at \( s \).

In the first step \( E(B) \) and \( E(R) \) are empty sets and \( V(B) = V(R) = \{ s \} \). \( E(B) \) and \( E(R) \), represent the sets of edges of out-branching \( B \) and \( R \), respectively. \( V(B) \) and \( V(R) \) represent the sets of vertices of out-branching \( B \) and \( R \), respectively. For each vertex \( u \) in \( G \) that is not simultaneous in \( V(B) \) and \( V(R) \) (\( u \in V(G)\backslash(V(B) \cap V(R)) \), the algorithm does the following:

First it finds a path \( P \) in \( G(V, E - E(B)) \). \( G(V, E - E(B)) \) is the pair of sets that composes graph \( G \), \( V \) represent the set of vertices and \( E - E(B) \) is the set of edges of \( G \) subtracted by the set of edges of \( B \). The algorithm then concatenates path \( P \) with out-branching \( R \).

![Fig. 10 – First iteration of algorithm 2: Part I (path \( P \) is added to out-branching \( R \))] (image)

After that the algorithm finds a path \( P' \) in \( G(V, E - E(R)) \). The algorithm then concatenates path \( P' \) with out-branching \( B \).

![Fig. 11– First iteration of algorithm 2: Part II (path \( P' \) is added to out-branching \( B \))] (image)

The union of \( P \) and \( P'^R \) is a closed walk, where \( P'^R \) represents the reverse links of \( P' \). Given that \( G \) is symmetric, we can use the reverse links of \( P' \) and add them to out-branching \( R \), excluding obviously the links that lead to vertices already in out-branching \( R \) (this does not happen in the first iteration) (Fig. 12).
To finish the iteration, the algorithm adds the reverse links of path $P$ to out-branching $B$, excluding, as before, the links that lead to vertices already in out-branching $B$ (Fig. 13).

This process is repeated until all vertices are part of both out-branchings.

The complexity of this algorithm is $O(|E(G)||V(G)|)$.

C. Algorithm 3: Finding a maximal set of link disjoint out-branchings in a graph

This algorithm calculates the maximum number of link-disjoint out-branchings in a given graph. The algorithm begins calculating the maximum number of possible link disjoint out-branchings. To do this the algorithm finds the minimum number of link disjoint paths between the root and every node of the graph.

Once the algorithm determines the maximum number of link-disjoint out-branchings rooted at $s$, it builds them one at a time. In turn, each of these out-branchings is build one link at a time starting from the root and stopping when we get an out-branching, a spanning out-tree. Before adding the $(k-i)^{th}$ link $e$ to an out-branching, the algorithm verifies if $i$ link-disjoint paths, from $s$ to the vertex where $e$ points to, will still be available after its addition. In case of success the link is added to the out-branching, otherwise it is marked and other link is chosen to be verified again. This process is repeated for every link to be in an out-branching for each out-branching to be calculated.

Let us see an execution example.

First algorithm 3 is used to calculate the maximum number of disjoint out-branchings $k$, that are available in graph above. In this case $k = 2$. After this the algorithm starts building the out-branchings, one at a time.

The first out-branching (out-branching[1]) is initiated with the root $s$. The algorithm then finds a link $e \in E(G) - R$ with $\text{tail}_G(e) \in \text{out-branching}[i]$ and $\text{head}_G(e) \in V(G) - V(\text{out-branching}[i])$. Where $\text{tail}_G(e)$ is the vertex to where link $e$ points to and $\text{head}_G(e)$ is the vertex from where link $e$ departs from. We say that link $e$ leaves $\text{head}_G(e)$ and enters $\text{tail}_G(e)$. An example of such link is represented in green in the figure below.
After choosing a link, the algorithm evaluates if there are at least \( k - i \) paths from \( s \) to \( \text{head}_G(e) \), that are simultaneously disjoint from \( e \), where \( k \) is the maximum number of out-branchings in graph \( G \) and \( i \in [1,k] \) is the variable that represents the current iteration. In this case \( k = 2 \) and \( i = 1 \). Given so, the algorithm needs to check if there is at least 1 path from \( s \) to \( \text{head}_G(e) \), that is disjoint from \( e \). In case of success the path is added to out-branching[1] and removed from \( G \), otherwise \( e \) is rejected, and therefore added to set \( R \), \( R = R \cup e \).

In this case, there was a disjoint path available as we can see in the above figure. Nevertheless, such is not always possible, as we can see in Fig. 18.

The algorithm continues to add links to the out-branching, until it spans all the vertices of \( G \). After that the iteration will be \( i = 2 \), and second out-branching is constructed with the links left. The two obtained out-branchings are presented below.

The complexity of this algorithm is \( (k^2 |E(G)|^2) \), where \( k \) is the maximum number or link-disjoint out-branchings rooted at \( s \).

**IV. COMPARISON OF OUT-BRANCHINGS**

**A. Main Out-Branchings**

To better understand the advantages and disadvantages of all the out-branchings generated by the algorithms we have seen, we have generated 100 random 2-connected symmetric graphs with 50 vertices each. To each link was attributed a random cost between 1 and 10. Note that two symmetrical links can have different costs in our simulation. We have then applied the different algorithms to the generated set of graphs and analyzed the resultant out-branchings.

The results of the presented algorithms will be compared with spanning SPTs that were also calculated for the generated graphs. In the case of algorithms 2 and 3, which generate two out-branchings, only the out-branching that has the lowest total link cost is contemplated.
The following table shows the results of the application of the algorithms that we have seen. We have focused our evaluation in 4 important points:

- Medium out-branching cost – the arithmetic mean of the sum of the costs of all the links in the out-branchings;
- Medium path cost – the arithmetic mean of the cost of the paths between the root and all the nodes in the out-branchings;
- Medium percentage of non-minimum paths – the arithmetic mean of the percentage of paths (between the root and each vertex) that have total link cost that is higher than the one provided by a SPT, on each out-branching;
- Resiliency – the granted possibility of having protection paths, that are disjoint from the main trees.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Out-branching Cost</th>
<th>Path Cost</th>
<th>Non-Minimum Paths</th>
<th>Resiliency</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT</td>
<td>201.7</td>
<td>6.66</td>
<td>X</td>
<td>NO</td>
</tr>
<tr>
<td>MOB</td>
<td>161.6</td>
<td>9.80</td>
<td>62.02%</td>
<td>NO</td>
</tr>
<tr>
<td>Algorithm 2 (Not improved)</td>
<td>236.5</td>
<td>8.94</td>
<td>41.54%</td>
<td>YES</td>
</tr>
<tr>
<td>Algorithm 2 (improved for the backup paths)</td>
<td>224.7</td>
<td>8.31</td>
<td>24.04%</td>
<td>YES</td>
</tr>
<tr>
<td>Algorithm 2 (improved for the main out-branching)</td>
<td>207.2</td>
<td>7.45</td>
<td>12.70%</td>
<td>YES</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>256.5</td>
<td>8.18</td>
<td>34.62%</td>
<td>YES</td>
</tr>
</tbody>
</table>

Table 1 - Out-branching comparison

The different algorithms offer advantages and disadvantages in different points. The minimum out-branchings (MOBs) offered of course the lowest total link cost. However the price to pay in terms of medium path cost is high. The total link cost of the path that goes from the root to each vertex is higher. In the point of view carrier, MOBs are the most efficient design in terms of link usage. MOBs use the design that offers the minimum total link cost. In the point of view of the users SPTs offer a lower root to vertex path cost.

Algorithms 2 and 3 grant resiliency. This means that it is always possible to find a link-disjoint backup path to any link of the main tree. We can see that the version of algorithm 2, that is optimized for the main tree, has in fact much better results. When compared to algorithm 3, the first seem to take advantage in all the compared points.

When optimized for backup paths, algorithm 2 has an allround performance. The cost of the backup paths is presented bellow.

As we have seen, algorithm 2 requires a 2-connected undirected graph. In practice, backbone networks assume this configurations, since they are in fact at least 2-connected and physical links offer communication in any direction, hence they are also symmetric.

When compared with the SPT, algorithm 2 shows almost no difference, only 12.70% of the paths are non-minimum paths. The total tree cost is 2.72% higher. Algorithm 2 is actually an excellent choice since it grants resiliency with only liminar extra cost.

### B. Protection Paths

Using multiple out-branchings is a way of protecting the network in case of link failure. We can choose to alternate packet distribution through the available out-branchings. However, we can also calculate backup paths to each link of the main out-branching, avoiding the need of a second out-branching. If we do this we can easily introduce FRR in the network, ending up with the fast resiliency advantages that FRR grants.

Both algorithms 2 and 3 offer resiliency because they calculate 2 or more link-disjoint out-branchings. If when finding the first out-branching, they grant that, at least another out-branching is available in the graph, they also grant paths from the root to each node, all of them disjoint from the main out-branching.

To make better use of FRR, every link of the out-branching should be protected and the beginning of the protection path should be located at the tail of the failed link. As far as algorithm 2 is concerned, the requirement was having a 2-connected symmetrical graph \( G \). In such graph, every link has its symmetrical; therefore each link of an out-branching that leads to a given node has a symmetrical link leading to the root.

If every node has a disjoint path leading back to the root, then the union of this path with the branches of a secondary out-branching, that lead to the respective node, form a backup path that is disjoint from the main out-branching \( R \). Given this, we know that after finding the main out-branching \( R \) in algorithm 2, we will still have at least one path non-overlapping disjoint path left. We can easily calculate the minimum path that grants these features by using a minimum paths algorithm in graph \( G(V, E - E(R)) \).

Regarding algorithm 3, the requirements were different; we just needed to have \( k \)-disjoint paths from the root the each node, in order for the algorithm to work. With this requirement we do not guarantee the existence of a path from the nodes to the root. If we want assure that the main tree calculated in algorithm 3 has disjoint protection paths available, we need to strict the requirements and ask for a symmetrical graph with \( k \)-disjoint paths from the root to each vertex. We can use the symmetrical links to make paths to the root. In this case, like in algorithm 2, we have guaranteed the existence of the disjoint protection paths.

We have compared our two algorithms in terms of cost of the protection paths. We have used the already generated 2-connected symmetric graphs and respective main out-branchings. We have calculated the minimum protection path of each link of the out-branching, i.e. given a link \( e \) in the out-branching, we have calculated the cheapest path that goes from \( \text{tail}(e) \) to \( \text{head}(e) \) and is simultaneously disjoint from the out-branching. The medium costs of these paths are shown below.
Algorithm 2, when optimized for the main tree, presents slightly lower values in comparison to algorithm 3 and again has points in its favor. Results are even better when it is optimized to obtain better backup paths; nevertheless, the price to pay to these paths is high. Since the use of this paths is mostly temporary (few seconds at most).

Table 2 – Protection path cost comparison

<table>
<thead>
<tr>
<th>Algorithm 2 (Improved for the backup paths)</th>
<th>Algorithm 2 (Improved for the main out-branching)</th>
<th>Algorithm 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.58</td>
<td>16.15</td>
<td>19.45</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS

In this thesis we presented ways of computing and implementing different out-branching designs, understanding the advantages and disadvantages of these implementations. Given a certain out-branching we purposed a protocol suite that could use this out-branching to transport IPTV packets and, at the same time, was capable of handling the requirements of IPTV. We have studied the interaction between the involved protocols and developed a strategy to operate them.

We have presented 3 different algorithms, which given a specific network, calculate different types of out-branchings. The algorithms had different objectives.

The first algorithm (algorithm 1) calculated the minimum cost out-branching (MOB). We have seen that a MOB is different from a shortest path out-branching and could not be calculated using a minimum cost spanning tree algorithm. In the given networks, MOBs are the out-branchings with lowest total link cost. However, they do not offer the shortest paths between the root and each vertex. In fact, they offered the highest path costs among the algorithms that we have compared. Algorithm 1 did not also guaranteed the existence disjoint paths that could substitute the links of the out-branching in link failure occasions.

The second algorithm (algorithm 2) generated two disjoint out-branchings given a 2-connected symmetric network. We have seen that in this type of network, after the calculation of the two out-branchings, it was possible to calculate local protection paths, for all the links in main out-branching, using the links left on the graph in association with the links of the backup out-branching. The protection paths were disjoint from the main out-branching. Given so, in case of link failure, traffic would not pass in the same link more than once, overloading the links. We have also used a heuristic to reduce the total link cost and the cost of paths that go from the root to each vertex in the out-branchings that result from algorithm 2. We have also calculated the protection paths for the main out-branching links and evaluated their costs.

Finally the last algorithm (algorithm 3) enabled the calculation of the maximal number of disjoint out-branchings in a given graph. We have demonstrated that if the network was symmetric, it was possible to calculate protection paths for main out-branching with the links left on graph.

When improved for the main out-branching algorithm 2 presented total links costs and root to vertex path costs that were similar to the ones obtained by an SPT. The costs were lower than the ones obtained by algorithm 3. As far as protection paths are concerned, its costs were also lower in algorithm 2 in comparison with algorithm 3.

One of the main problems of our implementations and that could lead to future work is that we are not using the Traffic Engineering properties of MPLS in its full extent. IP carriers use their backbone to support IPTV, but they also use it to support many other types of traffic. We are making our out-branching calculations on IGP costs which do not characterize the network properly. IGP costs, are usually based on the debit of the links and usually do not account for the current traffic of that link, as well as, the bandwidth that is being used at a given time. The paths that we assumed good (because of their low IGP cost) might not be that good if the links are being extensively use for other types of services like unicast data transmission or virtual private networks, for example. We are also not making use of constrained based routing offered by MPLS, which we could benefit from, if we wanted to balance the traffic over an IP backbone.

Nevertheless, the graph properties that led to development of algorithm show that in symmetrical networks we have some flexibility when choosing links to be part of the main out-branching, and also benefit from the resiliency of the protection paths that the algorithm accounts for. If we could find a set of metrics that dynamically characterize the network in all its extent, and in same time we could feed this data to algorithm 2 to calculate reconfigured versions of network, on the fly, we would certainly offer an interesting tool to enhance IP backbone networks.

ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to my supervisor Prof. João Luís Sobrinho for the continuous support of my M.Sc. study and research, for his patience, motivation and enthusiasm. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank my family, especially my parents, who supported me emotionally and financially all the way through the course.

Last but not the least, I would like to thank my beloved girlfriend for her patience to see me writing this thesis throughout the summer and during her vacations.

REFERENCES
