AnyTraffic routing algorithm for label-based forwarding

Pedro Pedroso*, Oscar Pedrola*, Dimitri Papadimitriou†, Miroslaw Klinkowski*‡, and Davide Careglio*
*Universitat Politécnica de Catalunya, Barcelona, Spain, Email: {pedroso, opedrola, careglio, mklinkow}@ac.upc.edu
†Alcatel-Lucent Bell NV, Antwerpen, Belgium, Email: Dimitri.Papadimitriou@alcatel-lucent.be
‡National Institute of Telecommunications, Warsaw, Poland

***Abstract*—The high capacity provided by packet-switched networks is supporting the proliferation of bandwidth intensive multimedia applications which require multicasting capability. As a consequence on today's networks, unicast and multicast traffic compete for shared resources where a router must maintain both unicast and multicast forwarding states. Pursuing a forwarding state reduction, in this paper we introduce the novel concept of AnyTraffic data group which consists of a group of nodes receiving both unicast and multicast traffic over the same single minimum-cost network entity. A novel heuristic algorithm is specifically defined to accommodate such data group and has been compared with the standard shortest path (SP) algorithm - the optimal case for unicast routing - and a classical Steiner tree (ST) heuristic algorithm - the optimal case for multicast routing. Exhaustive experiments have been performed to validate the proposed algorithm.**

I. Introduction

Within the next-generation packet-switched networks bandwidth-intensive multimedia applications such as HDTV, interactive teleconferencing, distributed data processing and video broadcasting are catching on. These new services and applications would save on network's capacity by multicasting information from a source to a set of destination nodes in addition to the conventional unicasting of information from a source to a single destination node. Therefore a mixed traffic scenario – where both traffic types may demand high bandwidth capacity – is the one more likely to be found in today's meshed aggregation environment. In order to cope with these new service requirements as well as reducing operational costs (CAPEX), Network Providers aim to evolve their infrastructure to a more efficient, scalable, and secure packet-switching infrastructure. Traffic-engineering architectures, such as multi-protocol label switching (MPLS) [1] and its generalization, (G)MPLS [2], have been deployed in the last past years reflecting this intention. Making use of the label switching concept these architectures enable a set of advanced TE capabilities to optimize the utilization of network resources (resource-oriented performance) and to enhance the QoS of traffic flows (traffic-oriented performance). From the control plane perspective, a label switching architecture allows the set up of point-to-point (P2P) and point-to-multipoint (P2MP) data paths using constraint-based routing schemes. A constraint-based routing algorithm computes an end-to-end path that fulfills a combination of a set of policy and QoS constraints (such as bandwidth and delay). Constraint-based routing implies explicit routing from the source of the path, i.e., the route to be followed by the data path is explicitly encoded in data path setup message. At the forwarding plane level, label switching allows fast forwarding as well as a distintion between unicast (one-to-one) and multicast (one-to-many) traffic using simpler header information. In this context, corresponding paths are instantiated as switched data paths.

In this paper, we focus on constraint-based routing schemes able to compute data paths allowing to forward an offered load consisting of combined unicast and multicast traffic in packet-switched meshed networks. Two routing approaches are commonly applied in current switched technologies, namely 1) the set up of a set of P2P switched data paths to encapsulate whether unicast or multicast traffic (i.e., multicast traffic is replicated as many times as the number of edge nodes processing multicast traffic); and 2) the set up of dedicated P2P switched data paths for unicast traffic forwarding and dedicated P2MP switched data paths for multicast traffic forwarding.

This paper proposes an innovative traffic routing approach, where the switched data paths enable, as much as possible, forwarding both unicast and multicast traffic together (i.e., over the same path). For this purpose, we introduce the concept of AnyTraffic data group which consists of a group of nodes receiving both unicast and multicast traffic over the same single minimum-cost network entity (e.g., tree). Note that this novel routing scheme is seamlessly applicable over any existing infrastructure, such as IP/MPLS, Ethernet or any packet-based switching technology [3] [4] when the following conditions are met i) at control plane level: root-initiated point-to-multipoint and source-initiated point-to-point switched data path (e.g., Resource ReSerVation Protocol-Traffic Engineering (RSVP-TE) or alike) and ii) at the forwarding plane level: capability to distinguish multicast from unicast traffic by inspecting other header information than the destination address (e.g., label/tag). The objective of this paper is to introduce a novel routing scheme adopting the new concept of AnyTraffic data group and compare it against two commonly used schemes. The aim of this novel scheme is to benefit from the advantages of the traditional approaches while minimizing their respective drawbacks and to achieve better system resource consumption (for state maintenance). Indeed, each switched data path is identified by an entry recorded in the forwarding table of each node (or forwarding state). To this forwarding table entry

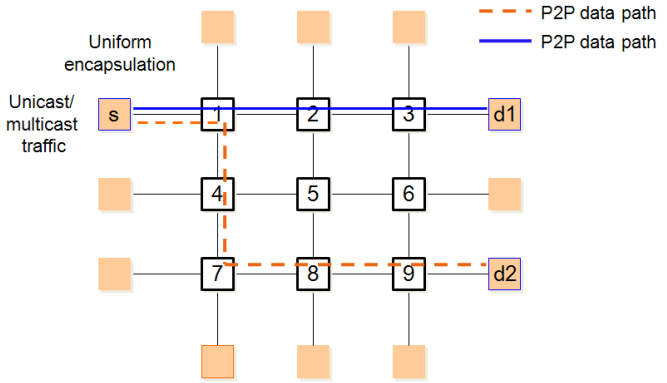

Fig. 1. Approach 1: both unicast and multicast traffic is carried over point-to-point data paths between each edge-node pair. In the example, $(s, d1)$ P2P data path and $(s, d2)$ P2P data path.

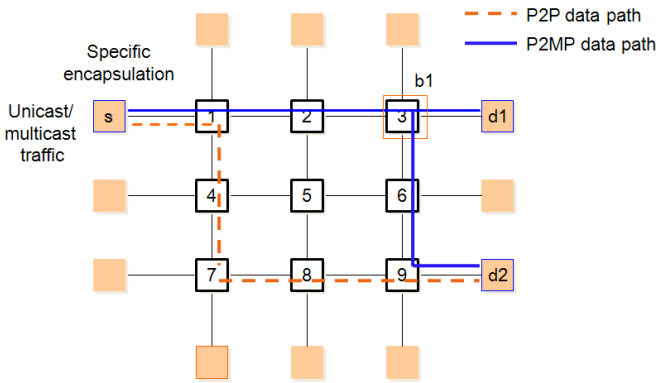

Fig. 2. Approach 2: multicast traffic is mapped to dedicated P2MP data paths and unicast traffic is mapped into dedicated P2P data paths. In the example, $(s, \{d1, d2\})$ P2MP data path, and $(s, d2)$ P2P data path.

corresponds at least one routing table entry that identifies (and maintains the state of) the path towards its destination (or routing state). Hence, a router may take a long time to look up the forwarding state for each arriving packet when a large number of multicast groups coexist. Indeed, state maintenance is one of the major problems that limits the scalability of any multicast deployment (e.g., [5]). With the introduction of AnyTraffic routing, we are pursuing the reduction of the total number of states needed to setup and maintain a data path by forwarding both unicast and multicast traffic together (i.e., over the same path, which implies the record of just one state) as much as possible. To meet this objective combined with the decrease of both bandwidth consumption and length of unicast path, a threshold has to be specified to decide where to separate the unicast from the multicast forwarding path (i.e., the placement of a branch node). In this study a novel heuristic algorithm is defined to accommodate the newly AnyTraffic data group and to find the proper set of branch nodes of the minimum-cost network entity.

This paper is organized as follows. Section II describes the three routing strategies considered on behalf of this study to forward an offered load of combined unicast and multicast traffic: the two traditional routing strategies and the outcome strategy introduced by this study. In Section III, the formulation of AnyTraffic routing problem is presented with some highlights on the needs for a novel heuristic algorithm for the proposed routing strategy. Section IV consists in a detailed formal formulation of the AnyTraffic routing algorithm, where a novel Steiner tree-based heuristic is specifically devised to accommodate the AnyTraffic data group. Section V presents the performance analysis based on extensive experiments over a set of network topologies. Conclusions are presented in Section VI.

II. Related Work and Contributions

In today's meshed networks relying on packet-switched technologies, multicast traffic can be carried by means of the following approaches.

The first approach (AP1) consists in setting up point-to-point data paths (referred to as network trunks) between each pair of source-destination nodes and forward both multicast and unicast traffic over these switched paths. (Constraint-based) shortest path algorithm is commonly used to compute the corresponding path across the network topology. This approach implies that the multicast traffic is replicated as many times as the number of destination nodes processing multicast traffic. This results in saving system resource for state maintenance at the expense of higher bandwidth consumption. For instance, as can be observed in Fig. 1, the two independent P2P data paths initiated at node s and with different destination nodes, namely $d1$ and $d2$, overlap at the first hop where, in the case of multicast traffic, the same information is transmitted twice over the same shared link.

The second approach (AP2) consists in setting up dedicated point-to-multipoint data paths for multicast traffic in addition to point-to-point data paths for unicast traffic. Steiner tree heuristics [6] are commonly used to construct the minimum cost tree dedicated to multicast traffic. In this case, source nodes have to provide for differentiated treatment of incoming native multicast vs. unicast traffic such as to map it in the corresponding data path. Multicast traffic is mapped to P2MP data paths and unicast traffic is mapped into P2P data paths (see Fig.2). P2MP data paths can be either inclusive or selective. Inclusive implies that a single P2MP data path is setup for the entire set of multicast groups to a set of edge nodes. Note that the set of edge nodes may be greater than the number of edge nodes of each individual multicast group resulting thus in saving state at the expense of bandwidth waste. Selective P2MP (the case considered in this study) implies that each multicast group is mapped into a dedicated P2MP data path, resulting thus in saving bandwidth at the expense of system resource needed for additional P2MP state maintenance. This approach also implies that dedicated point-to-point data paths must be provisioned for unicast traffic.

Being the computation of a minimum-cost Steiner tree an NP-complete problem [7], in this study we employ the minimum cost path heuristic algorithm (STH) [8] to compute

a minimum-cost Steiner tree for a P2MP data path. In STH, starting from a source node, the tree is gradually grown until it spans all destination nodes for the corresponding multicast group. The growth is usually based on the addition of shortest paths between destination nodes already in the tree and destination nodes not yet in the tree.

Traditionally, packet-switched technologies can carry multicast traffic using one of the above approaches. We propose the following novel approach (AP3). It can be seen as a refinement of AP1, i.e., single network entities are provisioned that carries (as much as possible) both unicast and multicast traffic. However, in this case, point-to-point data path segments are appended to designated branch nodes toward edge nodes that belong to a given set of one or more multicast groups. As shown in Fig. 3, both traffics are forwarded together over the same network entity till branch node $b1$. From there on, two appended P2P paths go toward the destination nodes $d1$ and $d2$ belonging to the AnyTraffic data group. Note that branch node $b1$ is not identical to the node resulting from the Steiner tree algorithm as applied in AP2.

The objective here is thus to benefit from the advantages of the traditional approaches while minimizing their respective drawbacks, i.e., keep the state maintenance overhead as low as possible while avoiding bandwidth waste by i) relying on replication of multicast traffic at branching points only (like in AP2) and ii) keeping unicast traffic transmission over "as short as possible" data paths (like in AP1).

To the best of our knowledge, there has not been any study suggesting a constraint-routing algorithm able to compute data paths allowing to forward unicast and multicast traffic together. Our contribution consists in i) designing and validating the algorithmic requirements to compute the path of the underlying data path structure (in the context of source-initiated routing) and ii) comparing the bandwidth and system resource consumption (in terms of state maintenance) for a given set of multicast groups against the traditional approaches.

It is worth to mention that this comparison is performed on different but stable scenarios where both multicast and unicast sessions are of infinite duration. Although not realistic (especially for multicast sessions), we emphasize that this work represents a first evaluation of the proposed AnyTraffic concept and the relative routing algorithm with the aim of encouraging further, more realistic developments.

III. Problem Formulation

The problem under investigation implies the conception of a novel heuristic algorithm to accommodate the proposed routing approach. Indeed, the direct application of one of the routing algorithms used by traditional approaches would result in an underperforming solution. For instance, if we apply the Shortest Path algorithm, we would be underperforming in terms of multicast routing (as commented in the former Section II for AP1). On the other hand, if we apply Steiner tree heuristic algorithm, we would be underperforming in terms of unicast routing because the path to carry the unicast traffic - towards one of the leaves of the multicast tree - would be too long when compared to the P2P shortest path, requiring thus additional bandwidth.

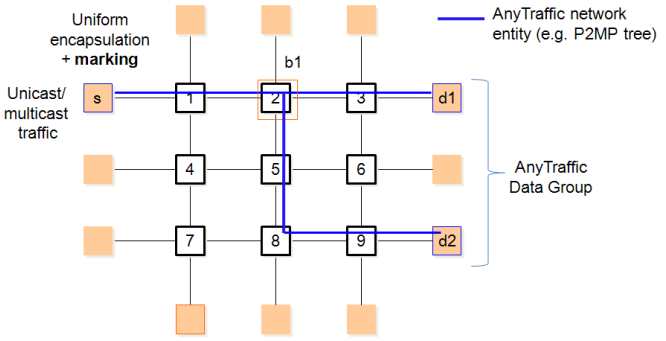

Fig. 3. The proposed AnyTraffic Approach: an AnyTraffic network entity (e.g., P2MP-tree) to carry both unicast and multicast traffic. Note that b1 might be different from b1 of the approach 2.

The concept of AnyTraffic data group is therefore introduced to define a group of destination nodes receiving unicast and multicast traffic over the same source-initiated network entity, the AnyTraffic tree (Fig. 3). The novel heuristic algorithm, which is mathematically described Section IV, attempts to construct such network entity per (set of) each AnyTraffic data group(s). Hence, the aim of the heuristic algorithm is to find, at the minimum-cost, a set of branch nodes that takes into account unicast and multicast traffic constraints. At designated branch nodes, several P2P data path segments are appended to reach the destination nodes that belong to a given set of (one or more) AnyTraffic data groups. The branch node selection is performed according to a given pruning condition (see the approach followed in Section IV) in order to guarantee a low increase of bandwidth consumption as well as of the length of unicast path. In fact, the network entity resulting from this algorithm is a root-initiated tree that can be provisioned using the technique described in [9].

In particular, our heuristic places itself between Shortest Path and Steiner tree algorithms, achieving better overall performance to forward an offered load consisting of combined unicast and multicast traffic in packet-switched meshed networks, as presented in Section V. In the next section, the mathematical description of the novel heuristic algorithm is presented.

IV. AnyTraffic Heuristic Algorithm

Consider a network modeled by a directed graph $G = (N, L)$, where N represents the set of nodes and L represents the set of links. Each link $l \in L$ has an associated cost $c(l)$; Section IV-A presents the method we use.

Let $p_{i,j}$ and $p_{i,k,j}$ both denote a path from node i to node j where k is an intermediate node, with $i \neq j \neq k$. Let $s \in N$ denote a source node and let $T_{s,M} = (N^*, L^*)$ be a connected subgraph without cycles (i.e., a tree) of $G = (N, L)$, source-initiated at s, and with the set of destination nodes $M \subseteq N^* \subseteq N$, $L^* \subseteq L$. Hereafter, M is referred to as the AnyTraffic data group, and $T_{s,M}$ as the AnyTraffic tree.

Let $\varphi_{s,M}$ denote a traffic request between source s and an AnyTraffic group M where $M \subseteq N \backslash \{s\}, M \neq \varnothing$. If $|M| = 1$, $\varphi_{s,\{d\}} = \varphi_{s,M}$ is a traffic request with a single destination node d and $M = \{d\}$, i.e., a request for a P2P data path to forward unicast traffic (one-to-one). Otherwise, it is a traffic request with multiple destination nodes $d_1, \ldots, d_{|M|}$, i.e., a request for a P2MP data path to forward multicast traffic (one-to-many). The objective of the AnyTraffic routing algorithm is to construct a tree $T_{s,M}$ for a given source s and AnyTraffic group M, such that it supports both unicast $\varphi_{s,\{d\}}, d \in M$, and multicast $\varphi_{s,M}$ traffic requests.

Depending on the type of incoming traffic request, different alternatives are possible at a given source node s. If the request is of a multicast traffic $\varphi_{s,M}$ and an AnyTraffic tree $T_{s,M}$ is available (i.e., because another multicast request $\varphi_{s,M}$ has been already served), the request is supported by $T_{s,M}$. Otherwise, the AnyTraffic routing algorithm is performed to establish a new AnyTraffic tree $T_{s,M}$. As detailed in Section IV-B, such tree is built up by successive choices of a branch node $n^* \in N$ that meets a set of given conditions. If a unicast traffic request $\varphi_{s,\{d\}}$ arrives, three different situations can occur: i) $d \in M$, and $T_{s,M}$ (with $|M| > 1$) is available, $\varphi_{s,\{d\}}$ is supported by $T_{s,M}$; ii) $d \in M$ but $T_{s,M}$ is not yet created, a shortest path must be setup; or iii) $d \notin M$, a shortest path must be setup.

The AnyTraffic routing algorithm comprises two phases, namely an initialization phase and a tree computation phase.

A. Initialization Phase

The initialization phase of the algorithm consists in defining parameters and assigning initial values to them. Since these parameters depend only on the network topology, this phase is performed once off-line and their values can be stored in nodes' memory.

The first parameter to be calculated is the cost $c(l) \in Z_+$ of each link $l \in L$. Several methods can be found in literature (e.g., [6]), below we present the method we adopted where the number of hops is used as tie breaker. Firstly, we find a uniform segmentation of the maximum distance link $L_{\max}$ into $Q_{\max}$ intervals, where $Q_{\max}$ is the maximal node degree in the network. For instance, for the maximum link distance of $100km$ and the maximal node degree of 5 we have the following intervals: $]0, 20]$, $]20, 40]$, etc. Then, for each network link l we find the corresponding interval number i such that $\frac{(i-1)L_{\max}}{Q_{\max}} < d(l) \leq \frac{iL_{\max}}{Q_{\max}}$, where $d(l)$ is the length of the link l –which must be known. Therefore, the link cost is calculated as:

$$c(l) = \left\lceil \frac{Q_{\max} + (i-1)}{Q(l)} \right\rceil, \tag{1}$$

where $Q(l)$ is the degree of the origin node of link l. Such assigned costs give preference to shorter links and their calculation involves a smoothing factor inversely proportional to the node degree.

Let $x_{i,j}$ denote the cost of the shortest path from node i to node j, $i \neq j$, computed by the Dijkstra algorithm link cost $c(l)$. Accordingly, $x_{s,d}, d \in M$, is the cost of the shortest path from source node s to destination node d. Among all path costs, we can find $c_{\max} = \max\{x_{i,j} : i, j \in N, i \neq j\}$, which corresponds to the shortest path of maximal cost in the network.

Let the function $\Psi(x) : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ be a function defined as:

$$\Psi(x) = x\left(1 + e^{-\frac{f(x)}{c_{\max}}}\right), \tag{2}$$

This function specifies the threshold for the maximum cost of a path that is alternative to the path of cost x. The function $f(x) : \mathbb{R}^+ \rightarrow \mathbb{R}$ is defined as $f(x) = \alpha x - \beta$, where parameters $\alpha, \beta \in]0, 1]$ define the shape of the threshold function. In particular, $\Psi(x_{s,d})$ limits the acceptable cost deviation of an alternative path $p_{s,d}$ from the path given by the Dijkstra algorithm. The maximum growth of $\Psi(x_{s,d})$ is achieved when $\alpha \rightarrow 0$ and $\beta \rightarrow 1$. After performing a number of experiments, setting $\alpha = 0.7$ and $\beta = 0.3$ gives acceptable values to initiate the algorithm.

Having defined $\Psi(x)$, we can compute the *maximum deficit factor* $\Delta^{s,d}_{\max}$ for each shortest path $p_{s,d}, d \in M$:

$$\Delta^{s,d}_{\max} = \Psi(x_{s,d}) - x_{s,d} = x_{s,d} e^{-\frac{f\left(x_{s,d}\right)}{c_{\max}}}. \tag{3}$$

This factor determines the acceptable cost increment of an alternative path againts the shortest path. In fact, it quantifies how much cost deviation is tolerable in order to forward both unicast and multicast traffic on that path without incurring too much damage compared to unicast traffic forwarding along the shortest path. Since $x_{s,d}$ and $c_{\max}$ depend only on the topology, $\Delta^{s,d}_{\max}$ can be computed off-line during the initialization phase of the algorithm and this value remains constant along the tree computation.

B. Computation Phase

This phase of the algorithm consists in the AnyTraffic tree computation itself which is a progressive and iterative process. In order to facilitate understanding of the following description, an illustrative example is depicted in Fig.4. A more detailed explanation of such example is presented in Section IV-C.

Let's define leaf as the tuple $\omega_{\upsilon,\Lambda} = \{\upsilon, \Lambda\}$, where $\upsilon \in N$ is a leaf seed and $\Lambda \subseteq M$ is a subset of the AnyTraffic group. We define Ω as the set of leaves remaining to be processed. At the beginning, this set comprises only the initial leaf, $\Omega = \{\omega_{s,M}\}$, where s is the seed root from where the computation is initiated and which comprises all destination nodes M. For instance, in the example of Fig. 4 (left-hand side), the initial leaf is $\omega_{s,M}$, $M = \{d_1, d_2, d_3\}$. We also define the initial graph $T_{s,M} = (\{s\}, \varnothing)$. For each $\omega_{\upsilon,\Lambda} \in \Omega$, the algorithm searches for a branch node $n^* \in N$ to be included in $T_{s,M}$ such that s is connected through n^* to a subset of nodes comprised in Λ. At each iteration step, an arbitrary leaf $\omega_{\upsilon,\Lambda}$ is pulled out from Ω to be processed. For this leaf, a set of candidate branch nodes A_ω is found. The set A_ω is

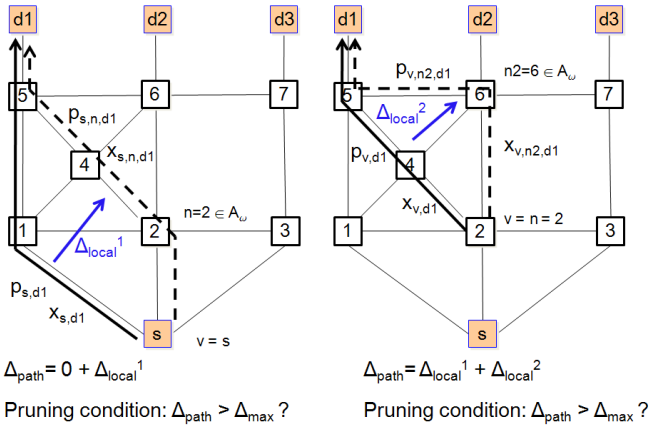

Fig. 4. Left-hand side: initial leaf $\omega_{s,M}$, $M = \{d1, d2, d3\}$; first step of the algorithm. Right-hand side: if previous pruning condition is satisfied and node 2 is selected as branch node, the process is repeat from the node 2 which is now the leaf seed of the leaf $\omega_{v,\Lambda}$, $\Lambda = \{d1, d2, d3\}$.

restricted to unvisited nodes in previous iterations that are adjacent to v and have the node degree equal or greater than three, i.e., the nodes that have at least two outgoing links, apart from the outgoing link to node v. In case the node degree of an adjacent node n is two, the first node with node degree equal or greater than three and laying on a path going from v through n is included into A_ω. For instance, in the example, $A_\omega = \{1, 2, 3\}$. The algorithm terminates when there is no leaves left in Ω and all destinations $d \in M$ can be reached from s in $T_{s,M}$.

At each candidate branch node $n \in A_\omega$, one alternative path $p_{v,n,d}$ is computed per destination $d \in \Lambda$, starting at v but forced to pass through n. A pruning condition determines if the set of alternative paths (one per each destination d) could be accepted, Indeed, each path $p_{v,n,d}$ may introduce higher cost $(x_{v,n} + x_{n,d})$ when compared to the cost $x_{v,d}$ of the shortest path $p_{v,d}$. Therefore, a *local deficit* factor $\Delta^{v,n,d}_{local}$ defined as

$$\Delta^{v,n,d}_{local} = (x_{v,n} + x_{n,d}) - x_{v,d}. \quad (4)$$

is calculated per each $d \in \Lambda$.

A *cumulative path deficit factor* for each $d \in \Lambda$, sums up, at node n, the local deficits produced by the alternative path $p_{s,n_1,\ldots,n_u,d}$ passing by already accepted branch nodes $n_1, \ldots, n_u$ of $T_{s,M}$ and new candidate branch node n:

$$\Delta^{s,d}_{path} = \sum_{i=0}^{u} \Delta^{n_i,n_{i+1},d}_{local} = \sum_{i=0}^{u} \left(x_{n_i,n_{i+1}}\right) + x_{n,d} - x_{s,d}, \quad (5)$$

where $n_0 = s$, and $n_{u+1} = n$.

Then, per each $d \in \Lambda$, a comparison between the cumulative path deficit (Eq. 5) and the maximum deficit factor (Eq. 3) is performed. If the maximum deficit constraint $\Delta^{s,d}_{\max} \leq \Delta^{s,d}_{path}$ is verified, i.e. if the cumulative deficit factor does not exceed the maximum deficit factor, the alternative path can be accepted. Otherwise, the algorithm removes node d from $\omega_{v,\Lambda}$ and creates a new leaf for further optimization.

When all candidate branch nodes in A_ω have been evaluated by running the pruning condition for each $d \in \Lambda$, branch node selection can be performed. The decision is taken by considering the minimum total deficit among all candidate branch nodes. Each of these nodes has its own deficit introduced by those $d \in \Lambda$ that meet the pruning condition. However, in order to reach fairness in the decision process, it is not enough for branch nodes to consider only the deficit based on the cost metric. Hence, we further ponder the deficit of each candidate branch node n at two levels: i) path level, by summing a fraction γ of the local deficit as defined in Eq. 4 to a fraction $(1-\gamma)$ of a local deficit also defined as in Eq. 4 but, instead of taking the cost metric, the calculus is performed using the hop count ($\Delta^{v,n,d}_{hop}$); ii) node level, a fraction σ of the ratio r, defined as the number of alternative paths meeting the pruning condition divided by the total number of paths that can reach all destinations (i.e. $|\Lambda|$), via $n \in A_\omega$. In summary, the *candidate deficit factor* $\Delta^{n,\omega}_{candidate}$ is defined as:

$$\Delta^{n,\omega}_{candidate} = \sum [\gamma \Delta^{v,n,d}_{local} + (1-\gamma)\Delta^{v,n,d}_{hop}] - \sigma r \quad (6)$$

In this study, after performing a number of experiments, we selected $\gamma = 0,5$ and $\sigma = 2$. These parameters can be further tuned in function of the bandwidth and state consumption objectives. The candidate branch node n^* that has the lowest candidate deficit (i.e., $n^* = \min\{\Delta^{n,\omega}_{candidate} : n \in A_\omega\}$) is selected as a branch node. Accordingly, $T_{s,M}$ is updated with all those links and nodes that lay on the path from v, which is the seed of the currently processed leaf $\omega_{v,\Lambda}$, to n^*.

At this point two new leaves can be available, namely, $\omega_1 = \{n^*, \Lambda_{n^*}\}$ (the leaf with the subset Λ_{n^*} of destination nodes that accepted $n*$ as a branch node) and $\omega_2 = \{v, \Lambda \setminus \Lambda_{n^*}\}$ (the leaf with the destination nodes removed by the pruning condition). Leaves ω_1 and ω_2 are added to the set Ω for further processing, respectively, if $|\Lambda_{n^*}| > 1$ and $|\Lambda \setminus \Lambda_{n^*}| > 1$. If either $|\Lambda_{n^*}| = 1$ or $|\Lambda \setminus \Lambda_{n^*}| = 1$ (i.e., only one destination remains in the subset), $T_{s,M}$ is updated with all links and nodes that lay on the shortest path, respectively, from n^* to $d \in \Lambda_{n^*}$ or from v to $d \in \Lambda \setminus \Lambda_{n^*}$. Note that the branch node n^* is excluded from the set of adjacencies of v, i.e., $A_{\omega_2} = A_\omega \setminus \{n^*\}$. Branch selection is repeated for each leaf left in Ω.

The pseudo-code of the AnyTraffic routing algorithm is given in Fig. 5.

C. Example

As illustrative example, Fig. 4 depicts two consecutive steps of the algorithm. The example shows the branch node evaluation mechanism to just one of the candidate branch nodes and to one destination node.

The left-hand side of the figure represents the initial step. For instance, consider the initial leaf $\omega_{s,M}$ where s is the node processing the incoming requests of both unicast and multicast traffic and the Anytraffic group $M = \{d1, d2, d3\}$. Node 2 is the current candidate branch node n being evaluated from a set $A_\omega = \{1, 2, 3\}$. This set corresponds to the adjacent nodes

INPUT: G, $c(l)$, $source\ s, M$
OUTPUT: $T_{s,M} = (N^*, L^*)$, $N^* \subseteq N, L^* \subseteq L$
INIT{
$\omega_{s,M} = \{s, M\}$
$\Omega = \{\omega_{s,M}\}$
$p_{s,d} \longleftarrow SP(s,d) \quad \forall d \in M$
$\Delta_{\max}(p_{s,d}) \quad \forall d \in M$
}
ANYTRAFFIC_HEURISTIC{
while $|\Omega| \neq \varnothing$
$\Delta_{candidate} = 0$
$\omega_{v,\Lambda} \longleftarrow \Omega, \quad \Omega = \Omega \setminus \{\omega_{v,\Lambda}\}$
$A_\omega = Adjacency\ nodes(v)$
for each $n \in A_\omega$
$\Lambda_n \longleftarrow Pruning\ Condition(n, \Lambda)$
$\Delta^{n,\omega}_{candidate} \longleftarrow Pdeficit$ (see Eq. (7))
end
$n^* = min\{\Delta^{n,\omega}_{candidate} : n \in A_\omega\}$
if $|\Lambda_{n^*}| > 1 \rightarrow \omega_1 = \{n^*, \Lambda_{n^*}\}$
else $\omega_1 = \varnothing$
if $|\Lambda \setminus \Lambda_{n^*}| > 1 \rightarrow \omega_2 = \{v, \Lambda \setminus \Lambda_{n^*}\}$
else $\omega_2 = \varnothing$
$\Omega = \Omega \cup \{\omega_1, \omega_2\}$
$T_{HEUR} = T_{HEUR} \cup p_{v,n^*}$
endwhile
}
Function $Pruning\ Condition(n, \Lambda)$**{**
for each $d \in \Lambda$
$p_{n,d} \longleftarrow SP(n,d)$
$\Delta_{path}(d)_k = \Delta_{path}(d)_{k-1} + \Delta_{local}(d)_k$
if $(\Delta_{path}(d) > \Delta_{\max}) \rightarrow \Lambda_n = \Lambda_n \setminus \{d\}$
endfor
return Λ_n
}

Fig. 5. The pseudo-code of the AnyTraffic heuristic algorithm being proposed.

of s with a node degree equal or higher than three. Thus, the computation of the local deficit, $\Delta^{s,2,d1}_{local}$, introduced by the alternative path, $p_{s,2,d1}$, is performed. Being the source node, the cumulative path deficit factor is equal to the local deficit. The next step is the verification of the pruning condition: if the cumulative path deficit $\Delta^{s,d1}_{path}$ is lower than the maximum deficit factor $\Delta^{s,d1}_{\max}$, the alternative path can be accepted and joined to a common trunk that supports the nodes remaining in Λ. Then, all candidate deficit factors of all accepted candidate branch nodes must be compared.

If we assume that the candidate branch node 2 is selected after all other candidates' evaluation, we observe the situation depicted in the right-hand side of Fig. 4. At this point, nodes s and 2 were added to $T_{s,M}$. The leaf seed is now $\omega_{2,\Lambda}$ and the same process described in the previous iteration is repeated. The set of candidate branch nodes is $A_\omega = \{1, 3, 4, 6\}$; node 6 and the alternative path $p_{2,6,d_1}$ are considered in the figure. The path deficit factor is now the sum of both local deficits, $\Delta^{s,2,d1}_{local} + \Delta^{2,6,d1}_{local}$. If it remains lower than the maximum deficit factor (which is constant along the algorithm computation), the new alternative path can be accepted. Otherwise, a new leaf must be created and added into the set of leaves Ω.

D. Complexity Analysis

The complexity of this algorithm is $O(|M| \cdot A \cdot H)$, where $|M|$ is the size of the AnyTraffic group (i.e., the number of destination nodes), A is the maximum node degree (i.e., the maximum number of adjacent nodes), and H is the hop distance between the source and the most remote destination node. This bound comes from the fact that at each hop towards the destination all adjacent nodes are checked as candidate branch node for destination nodes belonging to M. In a regular connected network ($A \ll |N|$) the complexity is low and, in any network, it might be reduced yet. The method consists in limiting the set of adjacent nodes that are within a given perimeter angle with respect to the next node belonging to the shortest path to every destination node. Preliminary results achieved by applying this method show no performance degradation while significant reduction of running time.

V. Performance Analysis

The performance of the proposed heuristic algorithm is evaluated and compared against the two strategies described in Section II in terms of bandwidth consumption and system resource consumption.

A. Scenario

In order to assess the robustness of the proposed solutions and determine the corresponding dependencies, several network topologies were simulated. Because similiar behavior was observed, we only present results for the following networks: Cost266 [10] and Rand37 with 37 nodes, as well as German50 [10] and Rand50 with 50 nodes. The differentiating properties of these topologies consist in different node degrees and clustering coefficient (c.c.) ranging in the interval [0,1], besides their respective number of nodes and links. Regarding the 37-nodes network topologies, the c.c. is 0.0 for the Cost266 and 0.2489 for the Rand37 topology. Regarding the 50-nodes topologies, the c.c. is 0.19 for the German50 and 0.2752 for the Rand50. Rand networks are produced by means of the algorithm proposed in [11] that randomly generates graphs from a sequence of node degrees. The traffic generation is a bound and discrete process. The unicast and multicast traffic are both generated within a bound range of two discrete traffic classes, namely class 1 - MPEG-4 standard definition (SD) - of 2 Mbps and class 2 - MPEG-4 high definition (HD) - of 8 Mbps.

Each node in the network is an ingress-egress node generating 150 connection requests. Different percentages of unicast and multicast traffic ratios are considered, namely 50%-50%, 75%-25%, and 95%-5%. For each multicast session, the size of the destination nodes ranges between a minimum of $log_2(N)$ and a maximum of $[log_2(N)]^2$, where N represents the set of nodes in the network. The simulations are performed in a non-blocking regime where enough network resource availability is assumed.

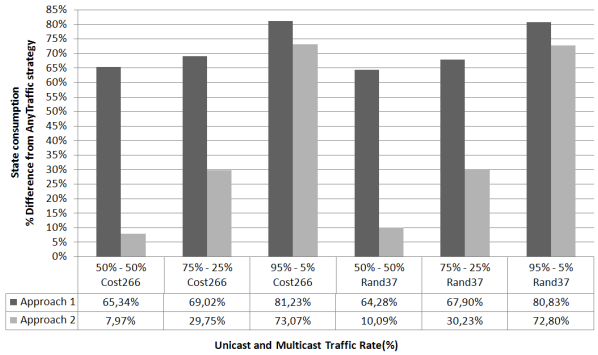

Fig. 6. State Consumption for the networks Cost266 and Rand37 of 37 nodes.

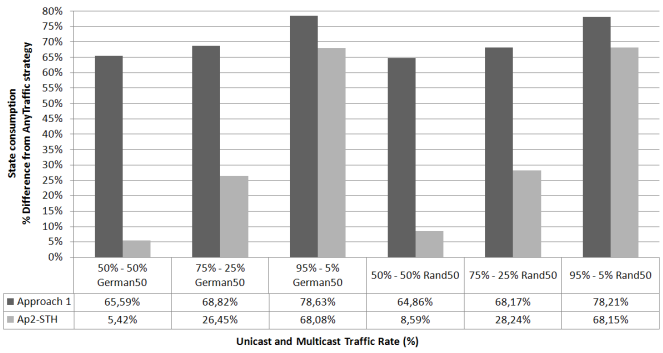

Fig. 8. State Consumption for the networks German50 and Rand50of 50 nodes.

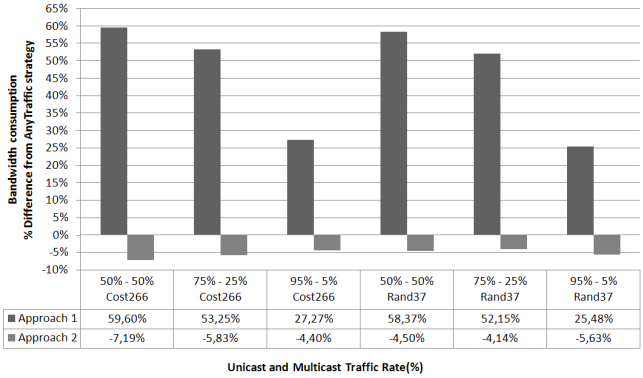

Fig. 7. Bandwidth Consumption for the networks Cost266 and Rand37 of 37 nodes.

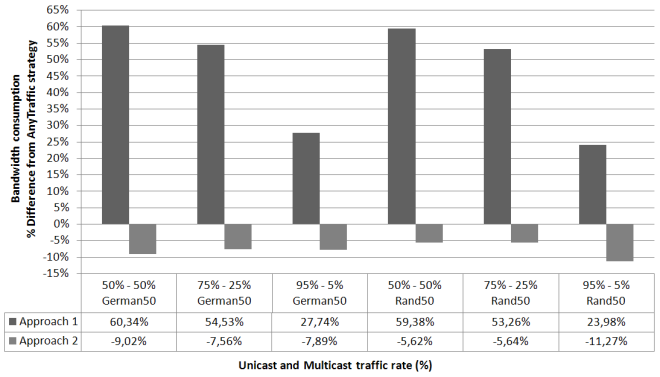

Fig. 9. Bandwidth Consumption for the networks German50 and Rand50of 50 nodes.

In order to settle the bounds of the algorithm, we have simulated a best case where all multicast requests are processed first. The simulation steps consist in i) creating the network entities (e.g. trees) for the AnyTraffic data groups, and then ii) processing the unicast requests looking for the minimum cost path among the created trees.

B. Results

We define the Relative Gain as the percentage in performance gain (in terms of either bandwidth or state consumption) achieved with the AnyTraffic routing scheme compared to the traditional routing schemes. The routing schemes taken as references are namely, the approach 1 and approach 2, detailed in Section II. The Relative Gain is formally defined as follows:

$$Relative\ Gain[\%] = \frac{APx - AP3}{APx} * 100 \qquad (7)$$

where the index $x = 1$ refers to the approach 1 (AP1) using SP algorithm to both traffic forwarding and the index $x = 2$ refers to the approach 2 (AP2) using SP algorithm to forward unicast traffic and Steiner Tree heuristic (STH) to forward multicast traffic. Therefore, the simulation results are displayed as the difference in percentage from AP1 and AP2 with respect to the Anytraffic approach, in function of the unicast and multicast traffic rate generated in the network.

Fig. 6 and 7 show the results obtained for the networks of 37 nodes, namely Cost266 and Rand37, in terms of state and bandwidth consumption, respectively. As it can be seen from these figures, the proposed approach (AP3) demonstrates outstanding performance in terms of state consumption compared to both AP1 and AP2 in the range of 50%-95% of unicast traffic rate. As unicast traffic rate increases, the gain increases. From Figures 6 and 8, for both pair of topologies (37 and 50 nodes), the state consumption gains range between 30% to 70% for the interval of 75%-25% to 95%-5% of traffic rate. Even though the network entities created by the multicast traffic requests are fewer, more unicast traffic uses them, reducing thus the number of states consumed compared to AP1 and AP2. In terms of bandwidth consumption, the AP3 shows worst performance due to the longer paths that unicast traffic has to follow. The observed bandwidth consumption tendency is inversely proportional to the state consumption. The additional bandwidth decreases because with less AnyTraffic entities created by multicast requests, forwarding unicast traffic requires more P2P (shortest) data paths. The number of P2P data paths becomes closer to the AP1 and AP2 values. However, this observation does not invalidate that a considerable amount of unicast traffic is still carried by means of AnyTraffic trees. Nevertheless, these values can be improved at the expense of decreasing a fraction of the state consumption gain by tuning the algorithm (e.g.

changing the values of γ and σ).

The same behavior is observed for the networks of 50 nodes, namely German50 and Rand50, shown in Figures 8 and 9. However, results are a bit less favorable compared to the results obtained with Cost266 and Rand37. This reflects that more nodes with higher node degree influence the performance of the algorithm. Although, the bandwidth consumption here is a little higher, regarding the considerable gains obtained for state consumption, we can decrease the bandwidth consumption again by tuning the algorithm to lower the aggregation of data paths. For instance, for the German50 network with 75%-25% of traffic rate, the AP3 results in a state consumption gainf od about 27% compared to AP2 versus an additional bandwidth consumption of about 8%. It is worth to note that for both pair of networks, the algorithm improves with respect to the network with the same number of nodes but lower clustering coefficient.

VI. Conclusion

This paper proposes a novel traffic routing scheme that aims at keeping the system resource consumption (in terms of forwarding state maintenance) as low as possible while limiting bandwidth consumption. This routing scheme relies on the concept of AnyTraffic data group that allows a group of destination nodes to receive both unicast and multicast traffic over the same source-initiated minimum-cost tree. A heuristic algorithm has been introduced that finds, per AnyTraffic data group, a single network entity (e.g., a tree) of minimum cost by taking into account both unicast and multicast traffic constraints for the selection of its branch nodes.

Two comparative approaches have been studied. The first (approach 1) relies on both unicast and multicast traffic forwarding over "as short as possible" data paths. The second (approach 2) approach makes use of shortest path routing for unicast traffic and the replication of multicast traffic at branch points of a tree computed by means of the minimum cost path algorithm, a Steiner Tree heuristic. To demonstrate the effectiveness of the proposed approach, extensive simulation experiments have been performed to compare the AnyTraffic routing algorithm against these two (more traditional) traffic routing approaches. The simulation results obtained are very satisfactory: it gives overall good improvements in state consumption while needs small increase in bandwidth utilization only againts approach 2. Therefore, further work is expected addressing issues like 1) dynamic multicast sessions where a edge node receiving traffic can be joined and released of a multicast group; 2) adapting the routing algorithm to optimize its performance when AnyTraffic trees partially overlap (where the same source-initiated multicast groups share almost the same nodes) to expectedly improve the state consumption results; and 3) study a blocking regime scenario and refine the proposed algorithm in order to achieve load-balancing and dynamical use of available bandwidth resources.

Acknowledgment

The authors would like to thank the grant SFRH / BD / 36950 / 2007 provided by the Portuguese Government Entity, Fundação para a Ciência e a Tecnologia (FCT). The work described in this paper was carried out with the support of the BONE-project ("Building the Future Optical Network in Europe"), a Network of Excellence funded by the European Commission through the 7th ICT-Framework Programme, and with the support of the COST 293 Action.

The authors would also like to thank Sébastien Rumley (EPFL, Switzerland) for the development of the Javanco software framework.

References

[1] E. Rosen, Ed., Multiprotocol Label Switching (MPLS) Architecture, *RFC 3031*, January 2001.

[2] E. Manie, Ed., Generalized Multi-Protocol Label Switching (GMPLS) Architecture, *RFC 3945*, October 2004.

[3] Z. Ortiz, G.N. Rouskas, H.G. Perros, "Scheduling of combined unicast and multicast traffic in broadcast WDM networks", in *Proc. PICS*, 1998.

[4] N. Singhal et al., "Optimal Multicasting of Multiple Light-Trees of Different Bandwidth Granularities in a WDM Mesh Network With Sparse Splitting Capabilities", *IEEE/ACM Trans. Net.*, vol. 14, no .5, Oct. 2006.

[5] T. Wong, R. Katz, "On analysis of multicast forwarding state scalability", in *Proc. ICNP 2000*, Osaka, Japan, Nov. 2000.

[6] F. K. Hwang, Ed., The Steiner Tree Problem, *Annals of Discrete Mathematics*, vol. 53, Amsterdam, North-Holland Editions, 1992

[7] M.R. Garey, R.L. Graham, D.S. Johnson, "The complexity of computing Steiner minimal trees", *SIAM J. Appl. Math.*, vol. 32, no. 4, pp. 835–859, 1977

[8] H. Takahashi, A. Matsuyama, "An approximate solution for the Steiner problem in graphs",*Math. Japonica*, pp. 573–577, 1980

[9] R. Aggarwal, Ed., Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs), *RFC 4875*, May 2007

[10] S. Orlowski, M. Pióro, A. Tomaszewski, R. Wesley, "SNDlib 1.0Survivable Neteork Design Library", in *Proc. INOC 2007*, Spa, Belgium, Apr. 2007.

[11] H. Kim, Ed., On realizing all simple graphs with a given degree sequence, *Discrete Mathematics*, Apr. 2008