Design and Performance Analysis of Dynamic Compact Multicast Routing

Dimitri Papadimitriou
Alcatel-Lucent Bell
Antwerpen, Belgium
dimitri.papadimitriou@alcatel-lucent.com

Pedro Pedroso, Davide Careglio
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain
{pedroso,careglio}@ac.upc.edu

Abstract—Recently introduced by Abraham et. al, compact multicast routing algorithms construct point-to-multipoint routing paths from any source to any set of destinations. The present paper introduces a different approach by proposing a name-independent compact multicast routing algorithm that is leaf-initiated, fully distributed, and independent from the underlying unicast routing. By means of the proposed routing scheme, resulting multicast distribution trees dynamically evolve according to the arrival of leaf-initiated join/leave requests. We provide the theoretical performance bounds of the proposed algorithm in terms of stretch of point-to-multipoint routing paths it produces, size and number of routing table entries, and communication/messaging cost. Next, we evaluate the performance of the proposed algorithm by simulation on synthetic power-law graphs (modeling the Internet topology) and the CAIDA map of the Internet topology. We also compare its performance to legacy multicast routing algorithms (the Shortest Path Tree and the Steiner Tree algorithm).

Keywords—component; multicast, compact routing

I. INTRODUCTION

Compact unicast routing aims to find the best tradeoff between the memory-space required to store the routing table (RT) entries at each node and the stretch factor increase on the routing paths it produces. Such routing schemes have been extensively studied following the model developed in the late 1980's by Peleg and Upfal [1]. Since then, in accordance to the distinction operated by Awerbuch [2] between labeled (nodes names are named by polylogarithmic size labels encoding topological information) and name-independent (node names are topologically independent) schemes, various compact routing schemes have been designed, notably in [3] and [4], respectively. These schemes are universal (they are designed so as to operate on any graph) however they are limited to point-to-point traffic (from a given source to a given destination).

As recently introduced by Abraham et al. in [5], dynamic compact multicast routing algorithms enable the construction of point-to-multipoint routing paths from any source to any set of destination nodes (or leaf nodes). The tree determined by a point-to-multipoint routing path is commonly referred to as a Multicast Distribution Tree (MDT) as it enables the distribution of multicast traffic from any source to any set of leaf nodes. By means of such dynamic routing scheme, MDTs can dynamically evolve according to the arrival of leaf-initiated join/leave requests. The routing algorithm creates and maintains the set of local routing states at each node part of the MDT. From this state, each nodes part of the MDT can derive the required entries to forward the multicast traffic received from a given source to its leaves.

In this paper, we propose a dynamic compact multicast routing algorithm that enables the construction of point-to-multipoint routing paths for the distribution of multicast traffic from any source to any set of leaf nodes. The novelty of the proposed algorithm relies on the locally obtained information (proportional to the node degree) instead of requiring knowledge of the global topology information (proportional to the network size). During the MDT construction, the routing information needed to reach a given multicast distribution tree is acquired by means of an incremental two-stage search process. This process, triggered whenever a node decides to join a given multicast source, starts with a local search covering the leaf node's neighborhood. If unsuccessful, the search is performed over the remaining unexplored topology (without requiring global knowledge of the current MDT). The returned information provides the upstream neighbor node along the least cost branching path to the MDT rooted at the selected multicast source node. The challenge consists thus here in limiting the communication cost, i.e., the number of messages exchanged during the search phase, while keeping an optimal stretch - memory space tradeoff.

To validate the design of our algorithm we determine the theoretical performance bounds in terms of i) the stretch of the point-to-multipoint routing paths it produces, ii) the memory space required to store the resulting routing table entries, and iii) the total communication or messaging cost, i.e., the number of messages exchanged to build the entire MDT. Note that the stretch is defined per [5] as the total cost of the edges used by the point-to-multipoint routing path (as produced by the routing algorithm) to reach a given set of destination or leaf set divided by the cost of the minimum Steiner tree for the same leaf set. Next, we evaluate by simulation the performance of the proposed algorithm by measuring the stretch of the point-to-multipoint routing paths, the size and the number of routing table entries as well as the communication cost. For this purpose, we simulate the execution of the proposed algorithm on synthetic power law graphs comprising 32k nodes that are representative of the Internet topology. We further compare the obtained performance results with the corresponding Internet CAIDA map. To contrast the actual gain obtained with the proposed algorithm, our performance analysis comprises on
one hand, the comparison between the results obtained for the proposed algorithm and those produced by the Abraham compact multicast routing scheme [5]. On the other hand, two reference schemes, the Shortest Path Tree (SPT) and the Steiner Tree (ST) algorithm are used to compare the performance of the proposed algorithm obtained when running over the same topologies.

This paper is organized as follows. Section II confronts our contribution to prior work on compact multicast routing while motivating the proposed approach compared to legacy multicast routing schemes. We detail the design of the proposed compact multicast routing algorithm in Section III and in Section IV, we provide the theoretical performance bounds in terms of the stretch of point-to-multipoint routing paths it produces, the memory space required for storing the routing table entries, and the communication cost. Section V analyses the performance results obtained by simulation over synthetic network topologies and CAIDA maps of 32k nodes. This section also compares the performance of the proposed algorithm with those realizable with the Abraham routing scheme. Finally, Section VI concludes this paper.

II. PRIOR WORK AND OUR CONTRIBUTION

Prior work on compact multicast routing is as far as our knowledge goes mainly concentrated around the schemes developed in the seminal paper authored by Abraham in 2009 [5]. One of the reasons we can advocate is that despite the amount of research work dedicated to compact unicast routing, current schemes are not yet able to efficiently cope with the dynamics of large scale networks. Therefore, running compact multicast routing independently of the underlying unicast routing system would be beneficial. This independence is even the fundamental concept underlying multicast routing schemes such as Protocol Independent Multicast [6]. Nevertheless, we also observe that the scaling problems already faced when unicast routing received main attention from the research community, remain largely unaddressed since so far. Indeed, multicast currently operates as an addressable IP overlay (Class D group addresses) on top of unicast routing topology, leaving up to an order of 100 millions of multicast routing table entries. Hence, the need to enable point-to-multipoint routing paths (for bandwidth saving purposes) while keeping multicast addressing at the edges of the network and build shared but selective trees inside the network. Indeed, in our approach, multicast forwarding relies on local port information only. Thus memory capacity savings comes from i) keeping 1:N relationship between network edge node and the number of multicast groups and ii) local port-based addressing for the local processing of multicast traffic. Further, we argue that compact multicast routing, by providing the best memory-space vs stretch tradeoff, can possibly address these scaling challenges without requiring deployment of a compact unicast routing scheme.

A. Preliminaries

Consider a network topology modeled by an undirected graph $G = (V,E,c)$ where the set $V$, $|V| = n$, represents the finite set of nodes or vertices (all being multicast capable), the set $E$, $|E| = m$, represents the finite set of links or edges, and $c: E \rightarrow Z^+$ that associates a non-negative cost $c(u,v)$ to each link $(u,v) \in E$. For $u, v \in V$, let $c(u,v)$ denote the cost of the path $p(u,v)$ from $u$ to $v$ in $G$, where the cost of a path is defined as the sum of the costs along its edges. Let $S, S \subset V$, be the finite set of source nodes, and $s \in S$ denote a source node. Let $D, D \subseteq V \setminus S$, be the finite set of all possible destination nodes that can join a multicast source $s$, and $d \in D$ denote a destination (or leaf) node. A multicast distribution tree $T_{s,d}$ is defined as an acyclic connected sub-graph of $G$, i.e., a tree rooted at source $s \in S$ with leaf node set $M, M \subseteq D$.

B. Prior Work

We outline the dynamic compact multicast routing scheme for join-only events scheme as designed by Abraham et al. (part of their seminal work [5]). In Section V, we provide a detailed comparison between the performance results obtained with our routing scheme and their approach. The Abraham scheme relies on the off-line construction of a bundle $B_k$ of sparse covers $TC_k, 2^k$, defined as $B_k = \{TC_{i,2^k}(G) | i \in I\}$ with $k = \log(n)$. Sparse covers are grown from a set of center nodes $c(T_{i,2^k})$ located at distance at most $k2^k$ from node $v$, where $T_{i,2^k}(v)$ denotes the tree in the collection of rooted trees $TC_{i,2^k}(G)$ that contains the ball $B(v,2^k)$. For each $i \in I$ and $T \in TC_{i,2^k}(G)$, the center node $c(T_{v})$ of each node $v \in T$ stores the labels of all nodes covered in the ball $B(v,2^k)$, the ball centered on node $v$ of radius $2^k$. Further, the SPLabel($v$) stores the label $\lambda(T,c(T))$ for each $T \in B(v)$, defined as set of all covers $T$ in the bundle $B_k$ such that $v \in T$. In addition, each node $v \in V$ stores the tree routing information $\mu(T,v)$ for all the trees in its own label SPLabel($v$). When a leaf node $u$ desires to join an MDT, it first determines whether or not one of the MDT nodes is already included in its local tree routing information table. If this is the case, the node sends a request to the center node $c(T_{v})$ with minimum degree $\mu(T_{v},v)$ so that is associated to that MDT node $v$. The center node $c(T_{v})$ then passes the label $\mu(T_{v},v)$, so that the selected MDT node $v$ can forward the multicast traffic to the newly joining leaf node $u$ (without further propagating this label to the source node $s$). Otherwise (some the leaf node covers define an empty intersection with the MDT), the leaf node $u$ with SPLabel($u$) queries the source node $s$ to obtain the set of MDT nodes it currently includes. Among all index $i \in I$, it then selects the tree $T_{i,2^k}(v)$ whose intersection with its bundle $B_k$ is minimum. Once the node, say $v$, part of this intersection is selected ($T_{i,2^k}(v) \in B_k(v)$), leaf node $u$ directs the join request to the associated center node $c(T_{i,2^k}(v))$. The latter passes a label $\mu(T_{i,2^k}(v),u)$ so that the selected MDT node $v$ can forward the incoming multicast traffic to the newly joining leaf node $u$. In order for the source node $s$ to reach node $u, v$ has to propagate the tuple $[v,\mu(T_{i,2^k}(v),u)]$ to source $s$. The leaf node $u$ updates all nodes covered by its balls $B(u,2^k)$ to allow them joining the MDT at node $u$.

\footnote{For simplicity, we present here the label-dependent variant of the scheme. In the name-independence version, center nodes store label mappings from names to nodes.}
As stated before, the reduction in memory space consumed by the routing table entries results however in higher communication cost compared to the reference algorithms, namely the SPT and the ST. Higher cost may hinder the applicability of our algorithm to large-scale topologies such as the Internet. Hence, to keep the communication cost as low as possible, the algorithm’s search process is segmented into two different stages. The rationale is to put tighter limits on the node space by searching locally in the neighborhood (or vicinity) of the joining leaf node before searching globally. Indeed, the likelihood of finding a node of the MDT within a few hops distance from the joining leaf is high in large topologies (whose diameter is logarithmically proportional to its number of nodes) and this likelihood increases with the size of the MDT. Hence, we segment the search process by executing first a local search covering the leaf node’s vicinity ball, and, if unsuccessful, by performing a global search over the remaining topology. By limiting the size (or order) of the vicinity ball taking into account the degree of the node it comprises, one ensures an optimal communication cost. For this purpose, a variable path budget \( \pi_r \) is used to limit the distance travelled by leaf initiated requests to prevent costly (in terms of communication) local search or global search. Additionally, as the most costly searches are resulting from the initial set of leaf nodes joining the multicast traffic source, each source constructs a domain (referred to as source ball). When a request reaches the boundary of that domain it is directly routed to the source.

## III. COMPACT MULTICAST ROUTING ALGORITHM

This section describes the design of the proposed compact multicast routing algorithm. We first provide an overall description of the proposed algorithm. Then, we detail the local and global search phases used to discover the least cost branching path from the joining leaf node to the MDT. We also specify the design of the on-line and distributed construction algorithm underlying the incremental least cost branching path discovery process.

### A. Description

The multicast distribution tree \( T_{s,D} \) is constructed iteratively. At each step \( \omega = 1, 2, \ldots, |D| \) of the leaf-initiated construction, a randomly selected node \( u \) joins \( T_{s,M} \), where \( M = D \) corresponds to the current set of nodes part of the MDT at a given construction step. If node \( u \) is already part of \( T_{s,M} (u \in V_T) \) then it is either a transit or a branching node of the MDT. Otherwise, node \( u \) is not part of \( T_{s,M} (u \in D \setminus V_T) \) and it must search for the least cost branching path from node \( u \) to node \( v \in T_{s,M} \). Among the set \( P_{u,v} \) of possible paths \( p(u,v) \) from node \( u \in T_{s,M} \) to node \( v \in T_{s,M} \), the least cost branching path \( p(u,v)^* \) is defined as follows:

\[
p(u,v)^* = \min\{c(u,v) | p(u,v) \in P_{u,v}\}
\] (1)

In this equation, the cost \( c(u,v) \) of the path \( p(u,v) \) is defined as the sum of the cost \( c(u,w) \) of the edge \( (u,w) \), where \( w = \text{succ}(u) \) refers to the upstream neighbor node of \( u \), and the cost \( c(w,v) \) of the path \( p(w,v) \). When each node along the least-cost branching path \( p(u,v)^* \) from leaf node \( u \) to \( v \in T_{s,M} \) determines its upstream neighbor node along that path, leaf node \( u \) can send to its selected upstream neighbor a request message to join \( T_{s,M} \). The join message is relayed along the selected least-cost branching path \( p(u,v)^* \) until it reaches node \( v \in T_{s,M} \). Once node \( u \) has joined \( T_{s,M} \), \( u \in V_T \), and the set \( M \) comprises node \( u \), we proceed to the next step by randomly selecting a node \( w \in D \setminus V_T \).

At the end of the iterative construction process, when all candidate leaf nodes have joined the MDT, i.e., \( M = D \) and \( D \setminus V_T = \emptyset \), \( T_{s,M} = T_{s,D} \). The routing table of each node \( v \in T_{s,M} (v \in V_T) \) includes i) one routing table entry (stored in the
multicast routing information base or MRIB) that indicates the upstream neighbor node to which the join message is sent for each source node s; this locally stored information enables performing Reverse Path Forwarding check so as to ensure loop-free forwarding of the incoming multicast packets, and ii) one multicast traffic routing entry (stored in the tree information base or TIB) to enable forwarding of incoming multicast traffic (generated from that source s) from its incoming port to a set of outgoing ports.

B. Least Cost Branching Path Discovery

Two types of messages are involved at each step of the least cost branching path discovery process, namely the request (type-R) messages flowing in the upstream direction towards the multicast source s, and the response (type-A) messages sent in the downstream direction towards the joining leaf node u.

- **Type-R message**: each message comprises the following information i) a sequence number \(\{u_{id}, r_{id}\}\) to prevent duplication of messages, where \(u_{id}\) identifies the leaf node \(u\) and \(r_{id}\) identifies its request to join the multicast source \(s\), ii) the leaf node \(u\)'s timer value \(\tau(u)\) that sets the waiting time at intermediates nodes before answering back to the downstream neighbor node, and iii) a path budget \(\pi\), starting at leaf node \(u\) from \(\pi(u) = \pi_{\text{max}}\) set at leaf node \(u\). The \(\pi_{\text{max}}\) value is bound by the graph diameter (the length of the longest shortest path) for which approximation algorithms exist, as well as method for computing a lower and upper bounds [8]. Starting from leaf node \(u\), the path budget \(\pi(u)\) is decremented at each node \(v\) according to the travelled distance: each traversed edge accounts for a distance decrease of 1. When the type-R message reaches node \(v\), if \(\pi(v) = 0\), the latter does not further propagate the type-R message in order to keep the communication cost as low as possible; otherwise, the value \(\pi(v)\) is decremented and passed to the neighboring nodes.

- **Type-A message**: sent in response to type-R messages, each type-A message comprises i) the cost \(c(w,v)^*\) of the locally selected least cost path \(p(w,v)^*\) from the local node \(w\) to \(v\) such that \(v \in T_{s,M}\); a node \(v \in T_{s,M}\) generating a type-A message to its downstream neighbor nodes sets this cost to infinite, and ii) when \(v \notin T_{s,M}\), the identifier of node \(v\).

Both types of messages comprise a dedicated tag, called flag_e, which enables distinguishing between messages exchanged during the search phases. Both type-R and type-A messages are tagged as internal when setting flag_e=0 (if belonging to the local search procedure), and as external when flag_e=1, otherwise.

1) Local Search

This first stage consists in a limited search within a certain perimeter around the joining leaf node \(u\). The contiguous set of nodes covered during this first stage is called the vicinity ball \(B \subset V\). Each node \(b \in B\) is therefore referred to as a vicinity node. The vicinity ball \(B\) of node \(u\), \(B(u)\), is delimited by vicinity edge nodes, \(b_i(u)\), i.e., nodes \(v \in V\) at a given hop-count distance from node \(u\).

At leaf node \(u\), the path budget \(\pi\) carried in the type-R message is initialized by setting its value \(\pi(u) = \pi_{\text{max}}\). If the degree of node \(u\), \(\deg(u)\), \(\deg(u) \leq \alpha\) then \(\pi(u) = 3\), if \(\alpha \leq \deg(u) < \beta\), \(\alpha < \beta\), then \(\pi(u) = 2\); otherwise \(\pi(u) = 1\). Values \(\alpha\) and \(\beta\) are small integer values determined a priori from the node degree distribution characterizing power law graphs. Starting from node \(u\), where \(\pi(u) = \pi_{\text{max}}\), the path budget \(\pi\) is decremented by 1 at each node \(v\) if \(\pi(v) < \beta\); otherwise it is set to 1. This condition prevents propagation of the type-R message to more than adjacent nodes of high degree nodes. At node \(v\), if the decremented \(\pi(v)\) value reaches 0, node \(v\) does not further propagate the type-R message in order to keep the communication cost as low as possible while increasing the likelihood of finding a node \(v \in T_{s,M}\). This procedure determines the maximum distance that type-R messages with flag_e=0 can traverse and determines the edge nodes of node's vicinity ball \(B(u)\). Indeed, vicinity edge nodes \(b_i(u)\) are the nodes for which the path budget \(\pi\) reaches 0.

When a given leaf node \(u\) decides to join the multicast source \(s\), it sends a type-R message to all the direct upstream neighbor nodes of node \(u\) (referred to as succ(u)) to find the least cost branching path \(p(u,v)^*\) to a node \(v \in T_{s,M}\) (\(v \in V_T\)). At condition that \(\pi(u)\) has not yet processed a type-R message with the same sequence number, \(\pi(u)\) successively propagates the message following a split horizon until it reaches either a node \(v \in T_{s,M}\) or a node \(v \notin T_{s,M}\) and \(\pi(v) = 0\). In the latter case, a vicinity edge node \(v\) is reached (node \(v = b_i\) but no node belonging to \(T_{s,M}\) can be found. Role of vicinity edge nodes is described in Section III.B.2.

At this point, node \(v\) replies to its downstream neighbor node(s) from which it has received the type-R message(s) with a type-A message. The type-A messages sent by node \(v\) in response to its downstream neighbor nodes \(w = \text{pred}(v) \in T_{s,M}\) are processed as follows. If node \(v = b_i \notin T_{s,M}\), then the type-A message (issued by node \(v\) to its downstream neighbors) sets the branching path cost to infinite. If not, then the type-A message (issued by node \(v\) to its downstream neighbors) sets this cost to value 0 which indicates that node \(v \in T_{s,M}\). Subsequently, each node \(w \neq b_i, v = \text{succ}(w)\), computes the branching path costs \(c(w,v)^*\) from itself to each node \(v\) by using (1), where either \(v \in T_{s,M}\) or \(v = b_i \notin T_{s,M}\). Node \(w\) then selects the least cost branching path \(p(w,v)^*\) and sends the corresponding cost value \(c(w,v)^*\) to its own downstream node(s) \(x\) such that \(w = \text{succ}(x)\). Observe that each node \(w\) maintains no additional routing information besides the degree(w) entries required at each step of the execution. At waiting timer \(\tau(u)\) expiration, if the set of type-A messages received by node \(u\) is empty or if the cost \(c(\text{succ}(u),v)\) is set to infinite in all received type-A message, node \(u\) declares the multicast source \(s\) unreachable. Otherwise, leaf node \(u\) determines among its neighbor nodes \(\text{succ}(u)\) from which it received type-A messages, the upstream node \(\text{succ}(u)^*\) along the least-cost branching path \(p(u,v)^*\) (= min\{\(c(u,\text{succ}(u)^*)\) + \(c(\text{succ}(u)^*),v\) | \(p(u,v) \in P_{a_{\text{id}}})\}) from node \(u\) to \(v \in T_{s,M}\). Node \(u\) then further proceeds by sending a message to \(\text{succ}(u)\) to join \(T_{s,M}\).
2) Global Search

This stage represents the search of the MDT's branching node outside the vicinity of the joining leaf node. This process is triggered by the leaf node $u$ when the local search phase declares the multicast source $s$ as unreachable in its vicinity ball $B(u)$. The global search phase is triggered by the leaf node $u$ and starts at each vicinity edge node $b(u)$. During this search phase, type-R and type-A messages tagged as external (i.e., $flag_e=1$).

In order to start a global search phase without restarting from the local neighborhood of the triggering node, the following procedures are considered:

- **The first procedure enables external type-R messages reaching the vicinity edge nodes without traveling along the complete set of nodes inside its vicinity ball $B(u)$.** For this purpose, the leaf node $u$ sends the external type-R messages directly to each of its vicinity edge nodes. Targeted forwarding of these messages from the leaf node $u$ to each vicinity edge node $b$ is possible because i) during the local search phase, the internal type-A messages (i.e., $flag_e=0$) received by the leaf node $u$ include the identifier of the node $b$ that initiates them, and ii) each vicinity node $b \in B(u)$ keeps per vicinity edge node $b(u)$ a single active interface from which type-A message with infinite cost has been received (indicating that the neighbor node sits along the path from leaf node $u$ to a given edge node $b$).

- **The second prevents that a given node $b \in B(u)$ receives back external type-R messages during the global search phase.** For this purpose, vicinity edge node $b$ filters incoming external type-R messages (i.e., $flag_e=1$). Remember that during the local search, internal type-A messages sent in response to the reception of type-R message ($flag_e=0$) are tagged with the $flag_e=0$. Interfaces sending such type-A message are removed from the list of interfaces for relaying type-R message ($flag_e=1$). The exception is for interfaces having received a type-R message ($flag_e=1$) with leaf node $u$ as sender to enable edge vicinity nodes to send back the answer to node $u$ once the global search completes for that node $b(u)$.

During the global search phase, the $\pi(u)$ budget value is set at node $u$ to a threshold equal to the graph diameter (length of the longest shortest path) and the waiting time $\tau(u)$ to a value that prevents waiting indefinitely. Moreover, upon reception of the type-R message ($flag_e=1$) from node $u$, each edge vicinity node $b(u)$ sets the maximum waiting timer $\tau(b(u))$ to $\tau(u) - 1$. The subsequent search process proceeds as follows: assume that node $b(u)$ sends an external type-R message to each of its upstream neighbor nodes except to its downstream node (part of the vicinity ball of the node from which the message has been received). At waiting timer $\tau(b(u))$ expiration, if the set of type-A messages received by node $b(u)$ is empty or if the cost $c(succ(b(u)),v)$ is set to infinite in all received type-A message, node $b(u)$ declares the multicast source $s$ as unreachable. Otherwise, node $b(u)$ determines among its neighbor nodes $succ(b(u))$ from which it received a type-A message, the upstream node $succ(v(u))$ along the least-cost branching path $p(v(u),v)$ to $T_{M,T}$ defined as:

$$p(b(u),v) = \min\{c(b(u),succ(b(u))), c(succ(b(u)),v)\} \quad \text{(2)}$$

Node $b(u)$ is ready to answer back to node $u$ once either of the following condition is met: i) it receives the entire set of type-A messages from its upstream neighbor nodes before its waiting timer $\tau(b(u))$ expires or ii) the waiting timer $\tau(b(u))$ initiated after reception of the first type-R message ($flag_e=1$) from leaf node $u$ expires. When one of these two conditions is met, node $b(u)$ selects the least cost branching path $p(b(u),v)$ and sends the corresponding cost value, $c(u,v)$ directly to the joining node $u$. At waiting timer $\tau(b(u))$ expiration, the set of type-A message received by node $b(u)$ is empty, the cost value $c(u,v)$ is set to infinite indicating that the multicast source $s$ is unreachable. Hence, as soon as this search phase completes, each node $b(u)$ returns a unique type-A message ($flag_e=1$) directly to the leaf node $u$ from which it initially received an external type-R message. Thus, contrary to the local search stage, no computation or selection is performed by nodes $b \in B(u)$ along the path taken by the type-A messages ($flag_e=1$) sent towards the leaf node $u$. This relay path is determined by the incoming interface maintained by each node $b \in B(u)$ upon reception of type-R message ($e=1$) from leaf node $u$. Note that the temporary records locally created during the local search phase are subsequently deleted by the node sending a type-A message ($flag_e=0$) that does not include an infinite cost to a vicinity edge node $b(u)$. The remaining records, locally created during the global search phase, are deleted by the node sending a type-A message ($flag_e=1$).

C. Source Node Vicinity Ball

As the most costly searches are resulting from the initial set of leaf nodes joining the MDT, each source constructs a source ball such that when a type-R message reaches the boundary of that domain it is directly routed to the source. This prevents searching at the neighborhood of the multicast traffic source. For this purpose, the multicast source node initiates a procedure that builds a ball around the size shall be at least as big as the average leaf node size. To be effective, this procedure shall construct a ball with i) size at least as large as the average size of leaf node's vicinity ball, and ii) radius computed from its outgoing ports shall be inversely proportional to the neighbor's node degree.

The procedure performs as follows: the source collects its neighbor's node degree to reach at a minimum size $x = x_{\text{min}}$ but up to a certain maximum diameter, $d_{\text{max}}$. At each collection step the following conditions are checked:

- If the value $x$ reaches the optimal value of the ball size $x_{\text{opt}}$ and diameter $d = d_{\text{opt}} \leq d_{\text{max}}$, then we get an optimal solution. The source ball construction stops.
- If $x > x_{\text{opt}}$ and the diameter $d$ being reached is such that $d < d_{\text{opt}}$, then the source node $s$ selects additional neighbor nodes so as to increase the diameter (up to value $d_{\text{max}}$) while limiting the increase above the value
bound analysis involves three different cases: a dynamic join scenario, the former is considered. The stretch cost globally leads to different stretch bounds. As we consider the cost of the edges used during the algorithm while building the entries, and iii) the total communication or messaging cost.

Thus, in total 2(#sourceBall nodes-1) additional routing table entries are to be considered. Note however that if a leaf node \( u \) located at a distance \( d(s,t) \) the nodes with smallest degree. Note that each node \( b_i(s) \in B(s) \), the vicinity ball of the source node \( s \), must now maintain an additional MRIB entry to relay type-R and type-A messages towards the source node \( s \). Thus, in total 2(#sourceBall nodes-1) additional routing table entries are to be considered. Note however that if a leaf node \( u = b_i(s) \in B(s) \), then that node \( u \) does not need any more to trigger any search procedure.

IV. THEORETICAL PERFORMANCE BOUNDS

In this section, we provide the theoretical performance bounds of the proposed algorithm in terms of i) the stretch of the point-to-multipoint routing paths it produces, ii) the memory space required to store the resulting routing table entries, and iii) the total communication or messaging cost.

A. Stretch

Minimizing the tree-cost sequentially, namely, the total cost of the edges used during the algorithm while building the multicast tree of the various stages and minimizing the tree-cost globally leads to different stretch bounds. As we consider a dynamic join scenario, the former is considered. The stretch bound analysis involves three different cases:

- Consider a joining node \( u \) and \( s \in B(u) \). Then the local search initiated by node \( u \) will find the least cost branching path if the path budget \( \pi(u) \) in the type-R message initiated is sufficient to reach the source node \( s \). It is obvious to see that if this condition is met, the resulting stretch increase is minimal.

- Consider a joining node \( u \) and \( s \notin B(u) \). If \( v \in T_{s,M} \) and \( v \in B(u) \), then the local search process will find the actual least cost branching path if and only if there no other node \( w \in T_{s,M} \) from the joining node \( u \) that can be found at shorter distance, i.e., \( d(u,w) \geq d(u,v) \). Indeed, the distance limit set by the joining node on the local search process by means of the path budget \( \pi(u) \) (see Section III), allows to reach a node \( v \in B(u) \) before triggering a global search. Due to the degree bound set when decreamenting the path budget \( \pi(u) \), a node \( v \in B(u) \) may be reached during the local search phase before reaching node \( w \in T_{s,M} \) and \( w \notin B(u) \) such that \( d(u,w) < d(u,v) \). Henceforth, the stretch increase is bound by the fraction of such nodes conditioned by the joining node selection and the current number of nodes \( \in T_{s,M} \). The stretch bound can thus be derived from the following formula:

\[
\frac{1}{N} \sum_{i=1}^{N} \frac{d(u,v_i)}{\min d(u,w_i)}
\]

where, \( \forall i \in M, [M] = N, \exists v_i, w_i \in V: \min d(u,w_i) < d(u,v_i) \) with a probability \( P(v_i \in T_{s,M} | v_i \in B(u)) \cdot P(w_i \in T_{s,M} | w_i \notin B(u)) > 0 \).

- Consider a joining node \( u \) and \( s \notin B(u) \). If \( v \in T_{s,M} \) and \( v \notin B(u) \), then the global search process will find the least cost branching path if the path budget \( \pi(u) \) in the type-R message initiated is sufficient to reach the source node \( s \). It is obvious to see that if this condition is met, the resulting stretch increase is minimal.

B. Storage

Each node \( v \in T_{s,M} \) stores in its local routing table at most one MRIB entry and at most one TIB entry whose size is proportional to the local tree out-degree \( k \) (as this entry indicates the outgoing ports for the incoming multicast traffic). Assuming a minimum port encoding proportional to \( \log(k) \), the storage size per node \( v \in T_{s,M} \) is \( O(k \log(k)) \) bits. Nodes \( b_i(s) \in B(s) \) require an additionally storage capacity for the MRIB enabling to relay type-R and type-A messages.

C. Communication

Each join event as initiated by a node \( u \), results in a communication cost \( C(u) \), i.e., the number of messages exchanged for node \( u \), to join a node \( v \in T_{s,M} \) that is given by the following formula:

\[
C(u) = 2m'X + 2m'Y + 2(m' - m')Z
\]

In this equation, \( m \) and \( m' \) are respectively the number of edges in the vicinity ball \( B(u) \) of node \( u \) and the total number of edges \( |E| \). \( X \) is the probability that at least one node \( v \in T_{s,M} \) is comprised in the ball \( B(u) \) of the selected node \( u \in V \). \( Y \) is the probability that none of the tree nodes \( v \in T_{s,M} \) are comprised in the ball \( B(u) \) of the selected node \( u \in V \). \( Z \) is the probability that all nodes \( v \in V \) are not in the ball of the selected node \( u \) (all tree nodes lie outside of the ball \( B(u) \) of the selected node \( u \)). The total communication cost, i.e., the cost to build the entire MDT is then determined by the sum of the individual communication costs \( C(u_i) \) of the \( i = 1, \ldots, |N| \) leaves composing the tree. In [7], we demonstrate that the derivative of (4) with respect to the number of nodes in the vicinity of node \( u \), \( |B(u)| \), provides the value of the ball size that minimizes the communication cost while corresponding to the order of the largest connected subgraph of diameter \( d_{max} \) that can be constructed.

V. PERFORMANCE ANALYSIS

The performances of the proposed compact multicast routing algorithm are analyzed by means of simulation on large scale topologies generated by GLP [9] and Internet CAIDA maps. The GLP evolutive topology model, which relies on generalized linear preferential attachment, produces power law graphs that are representative of the Internet Autonomous System (AS) topology (one node models an AS), in particular, in terms of clustering coefficient. The properties of these topologies are summarized in Table I.

The execution scenario considers the construction of point-to-multipoint routing paths for leaf node set of increasing size
from 500 to 4000 nodes (selected randomly) with increment of 500 nodes. Each execution is performed 10 times by considering 10 different multicast sources. We compare the performance of our algorithm to the Shortest-Path Tree (SPT) and the Steiner Tree (ST) algorithms.

- The SPT algorithm provides the reference for the communication cost. It is constructed from a loop-avoidance path-vector routing algorithm carrying the identifier of the multicast source s and the routing path to reach that source. Each node keeps thus a routing table (RT) entry per neighbor node (to exchange messages) and a RT entry per path to the source s.
- The ST algorithm provides the reference in terms of stretch. In order to obtain the near optimal solution for the ST, we consider a ST-Integer Linear Programming formulation. For this purpose, we adapted the formulation provided in [10] for bi-directional graphs. The communication cost for the ST measures at each step of the MDT construction the number of messages initiated by nodes part of the MDT. These messages contain the minimal information for remote nodes not (yet) belonging to the MDT to join it. Using this information, each node knows how to reach the closest node of the MDT. Thus, although the ST is computed centrally, the communication cost accounts for the total number of messages exchanged during the MDT building process as a dynamic scenario would perform.

### Table I: Topology Properties

<table>
<thead>
<tr>
<th>Topology Property</th>
<th>GLP</th>
<th>CAIDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes - Links</td>
<td>32618 - 146816</td>
<td>32000 - 120436</td>
</tr>
<tr>
<td>Avg - Max. Node Degree</td>
<td>4,50 - 2520</td>
<td>7,53 - 1165</td>
</tr>
</tbody>
</table>

### A. Stretch

This section details the simulation results obtained for the multiplicative stretch defined as the cost ratio between the point-to-multipoint routing paths (underlying the MDT) produced by the proposed scheme and the minimum Steiner Tree. We also compare the cost ratio between the point-to-multipoint routing path produced by the SPT and the minimum Steiner Tree.

1) **GLP Topology**

As shown in Figure 1, the multiplicative stretch for the proposed algorithm is slightly higher than 1 for the GLP topology. As the leaf node set increases from 500 to 4000, its trend curve decreases from 1.09 (maximum value reached for 500 leaf nodes) to 1.03 (minimum value reached for 4000 leaf nodes). Compared to the SPT stretch, our algorithm maintains a maximum deterioration of 4% for sets of 500 leaf nodes; this deterioration becomes negligible as the size of the leaf node sets increases.

2) **CAIDA Map**

As shown in Figure 2, the multiplicative stretch for the proposed algorithm is slightly higher than 1 for the GLP topology. As the leaf node set increases from 500 to 4000, its trend curve decreases from 1.08 (maximum value reached for 500 leaf nodes) to 1.05 (minimum value reached for 4000 leaf nodes). Compared to the SPT stretch, our algorithm maintains an average gain of 4% along the different group sizes.

### B. Storage

This section details the simulation results obtained for the memory capacity required to store the routing tables entries (underlying the MDT) produced by proposed scheme. It also determines the relative gain we obtain in terms of the ratio between the total number of RT entries produced by our algorithm and the total number of RT produced by the reference algorithms. This ratio provides an indication of the achievable reduction in terms of the memory capacity required to store the routing table entries produced by these algorithms.

#### 1) GLP Topology

From Table II, we can observe that the proposed algorithm (Our) produces significantly less RT entries than the ST and SPT references. The highest number of RT entries is obtained for a set of 4000 leaf nodes: 10154 RT entries. This value is 4,10 times smaller than the number of RT entries produced by the ST algorithm (41643 RT entries) and 27,92 times smaller than the number of the RT entries produced by the SPT algorithm (283477 RT entries).

Figure 3 illustrates the relative gain expressed in terms of the ratio between the total number of RT entries produced by the ST and the SPT references and our algorithm. An increasing gain can be observed as the size of the leaf node set decreases from 4,10 (leaf set of 4000 nodes) to 20,34 (leaf set of 500 nodes) compared to the ST algorithm and from 27,92...
(leaf set of 4000 nodes) to 166,08 (leaf set of 500 nodes) compared to the SPT algorithm.

### Table II: Number of RT entries for the SPT, ST, and Our algorithm with respect to the leaf node set size.

<table>
<thead>
<tr>
<th>Leaf Node Set Size</th>
<th>Routing Scheme</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SPT</td>
<td>ST</td>
<td>Our</td>
</tr>
<tr>
<td>500</td>
<td>274555</td>
<td>33483</td>
<td>1646</td>
</tr>
<tr>
<td>1000</td>
<td>275927</td>
<td>34733</td>
<td>2986</td>
</tr>
<tr>
<td>1500</td>
<td>277261</td>
<td>35965</td>
<td>4272</td>
</tr>
<tr>
<td>2000</td>
<td>278561</td>
<td>37191</td>
<td>5554</td>
</tr>
<tr>
<td>2500</td>
<td>279827</td>
<td>38349</td>
<td>6756</td>
</tr>
<tr>
<td>3000</td>
<td>281045</td>
<td>39443</td>
<td>7874</td>
</tr>
<tr>
<td>3500</td>
<td>282265</td>
<td>40559</td>
<td>9022</td>
</tr>
<tr>
<td>4000</td>
<td>283477</td>
<td>41643</td>
<td>10154</td>
</tr>
</tbody>
</table>

Figure 3. RT Size Ratio as a function of Leaf Node Set Size

2) **CAIDA Map**

From Table III, the proposed algorithm (Our) produces significantly less routing table entries that the ST and SPT reference algorithms. The highest number of RT entries is obtained for set of 4000 leaf nodes: 13169 RT entries. This value is 3.21 times smaller than the number of RT entries produced by the ST algorithm (42277 RT entries) and 14.38 times smaller than the number of the RT entries produced by the SPT algorithm (189431 RT entries).

### Table III: Number of RT entries for the SPT, ST, and Our algorithm with respect to the leaf node set size.

<table>
<thead>
<tr>
<th>Leaf Node Set Size</th>
<th>Routing Scheme</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SPT</td>
<td>ST</td>
<td>Our</td>
</tr>
<tr>
<td>500</td>
<td>180993</td>
<td>34111</td>
<td>4919</td>
</tr>
<tr>
<td>1000</td>
<td>182339</td>
<td>35391</td>
<td>6237</td>
</tr>
<tr>
<td>1500</td>
<td>183609</td>
<td>36609</td>
<td>7471</td>
</tr>
<tr>
<td>2000</td>
<td>184825</td>
<td>37779</td>
<td>8653</td>
</tr>
<tr>
<td>2500</td>
<td>186009</td>
<td>38935</td>
<td>9807</td>
</tr>
<tr>
<td>3000</td>
<td>187151</td>
<td>40047</td>
<td>10921</td>
</tr>
<tr>
<td>3500</td>
<td>188325</td>
<td>41199</td>
<td>12059</td>
</tr>
<tr>
<td>4000</td>
<td>189431</td>
<td>42277</td>
<td>13169</td>
</tr>
</tbody>
</table>

Figure 4 illustrates the relative gain expressed in terms of the ratio between the total number of RT entries produced by the ST and SPT references and our algorithm. An increasing gain can be observed as the size of the leaf node set decreases from 3.21 (leaf set of 4000 nodes) to 6.93 (leaf set of 500 nodes) compared to the ST algorithm and from 14.38 (leaf set of 4000 nodes) to 36.79 (leaf set of 500 nodes) compared to the SPT algorithm. Interestingly, the obtained gain values for the CAIDA map are smaller than those obtained for the GLP topology. This difference can be explained resulting from the difference in tree-depth: 6 (leaf set of 500 nodes) to 9 (leaf set of 4000 nodes) for the CAIDA map vs 8 (leaf set of 500 nodes) to 11 (leaf set of 4000 nodes) for the GLP topology.

C. **Communication Cost**

The communication cost is a critical metric to determine the applicability of the proposed compact routing algorithm to large scale topologies comprising as in the present work, i.e., 32k nodes. The two-stage search procedure and the source node vicinity ball construction, both presented in Section III, play an important role in mitigating the communication cost.

1) **GLP Topology**

As depicted in Figure 5, the communication cost ratio for the proposed algorithm is relatively high compared to the SPT even if much lower than the communication cost implied by the ST (not represented in this figure). Indeed, the communication cost ratio increases from 2.69 (leaf set of 500 nodes) to 8.17 (leaf set of 4000 nodes). This observation can be explained by the presence of high degree nodes (nodes that have a degree of the order to 100 or even higher) in power law graphs. However, as computed this communication cost does not take into account for the evolution of the routing topology. This evolution impacts multicast routing algorithms such as the SPT that are strongly dependent on non-local unicast routing information compared to the proposed algorithm. Moreover, as shown in Figure 5, the communication cost of the proposed algorithm compared to the SPT communication cost, decreases as the number of nodes composing the leaf node set increases. This trend leads us to expect that a saturation level can be reached around a communication cost ratio not higher than 10 to 15 as the size of the lead node set continues to grow. It is worth mentioning that the memory and the capacity required to process communication messages are relatively limited.

2) **CAIDA Map**

As depicted in Figure 5, the communication cost ratio for the proposed algorithm is relatively high compared to the SPT even if much lower than the communication cost implied by the ST (not represented in this figure). Indeed, the communication cost ratio increases from 2.69 (leaf set of 500 nodes) to 8.17 (leaf set of 4000 nodes). This observation can be explained by the presence of high degree nodes (nodes that have a degree of the order to 100 or even higher) in power law graphs. However, as computed this communication cost does not take into account for the evolution of the routing topology. This evolution impacts multicast routing algorithms such as the SPT that are strongly dependent on non-local unicast routing information compared to the proposed algorithm. Moreover, as shown in Figure 5, the communication cost of the proposed algorithm compared to the SPT communication cost, decreases as the number of nodes composing the leaf node set increases. This trend leads us to expect that a saturation level can be reached around a communication cost ratio not higher than 10 to 15 as the size of the lead node set continues to grow. It is worth mentioning that the memory and the capacity required to process communication messages are relatively limited.
The same trend can be observed for the CAIDA Map where the communication cost ratio between our scheme and the SPT algorithm increases from 7.88 (leaf set of 500 nodes) to 13.77 (leaf set of 4000 nodes). The difference observed between the CAIDA map and the GLP topology can be explained from the following observation the tree-depth differs by a unit (3 vs 4). This difference induces a relatively higher cost of the SPT when running over the GLP topology.

For the scheme allowing only dynamic join events, the performance in terms of the stretch of the point-to-multipoint routing paths produced and the memory space required by the proposed algorithm and by the Abraham scheme as specified in [5] (for dynamic join only events) are compared.

1) Stretch

For the scheme allowing only dynamic join events, the MDT cost is given by Lemma7 of [5]. The authors determine that the proposed dynamic multicast algorithm is $O(\min\{\log n, \log \Delta, \log n\})$ competitive compared to the cost of the optimal algorithm – Steiner Tree. In this formula, the factor $\Delta$ is the aspect ratio defined as the ratio between max $d(u,v)$ and min $d(u,v)$, for any $u, v \in V$. Considering an aspect ratio $\Delta$ of 6 and a network of 32k nodes the stretch is about 3.5. Thus the stretch upper bound of the point-to-multipoint routing path produced by the Abraham scheme, even if universal (applicable to any graph), is about 3 times higher than the one produced by our scheme.

2) Storage

Following the description of the Abraham scheme provided in Section II.B, the storage requirement is given by the memory space that includes:

- the tree routing information $\mu(T,v)$ stored by each node $v$, for all trees in its own SPT($v$) leading to a total storage of $O(\log n \log \Delta \log \log n)$ bits, ii) for each $i \in I$ and $T \in T(G)$, the center node $c(T(v))$ of each node $v \in T$ that stores the labels of all nodes contained in the ball $B(v,2^{i+1})$ leading to a total storage over all radii of $O(kn^{1+1/k} \log \Delta)$ bits; in addition, each node $v$ stores $O(\log \Delta)$ labels of size $O(kn^{1/k})$ each leading to a total memory consumption of $O(kn^{1/k})$ bits. The resulting memory storage requires about 700kbits for a tree comprising 4000 leaf nodes. For the same leaf set size, our routing scheme requires about 1250kbits.

VI. CONCLUSION

This paper introduces the first known name-independent compact multicast routing algorithm enabling the leaf-initiated, distributed and dynamic construction of MDT. The performance obtained shows substantial gain in terms of the number of RT entries compared to the ST (minimum factor of 3.21 for sets of 4000 leaf nodes, i.e., 12.5% of the topology size) and the memory space required to store them. The stretch deterioration compared to the ST algorithms ranges between 8% and 3% (for multicast group size of 500 to 4000, respectively); thus, decreasing with increasing group sizes. The proposed two-phase search process - local search first covering the leaf’s node vicinity, and if unsuccessful, a global search over the remaining topology - combined with the vicinity ball construction at the source node enables to keep the communication cost of the proposed algorithm within reasonable bounds compared to the reference SPT scheme and sub-linearly proportional to the size of the leaf node set. Future work will determine if these promising performance results can still be verified for dynamic sequences of node join and node leave events and non-stationary topologies.

REFERENCES