Compact Multicast Routing

Ittai Abraham[1], Dahlia Malkhi[1], and David Ratajczak[2]

[1] Microsoft Research, Silicon Valley Center.
ittaia@microsoft.com, dalia@microsoft.com
[2] Computer Science Department, UC Berkeley.
dratajcz@eecs.berkeley.edu

Abstract. In a distributed network, a *compact multicast scheme* is a routing scheme that allows any source to send messages to any set of targets. We study the trade-off between the *space* used to store the routing table on each node and the *stretch* factor of the multicast scheme – the maximum ratio over all sets of nodes between the cost of the multicast route induced by the scheme and the cost of a steiner tree between the same set of target nodes. We obtain results in several variants of the problem: *labeled* – in which the designer can choose polylogarithmic node names, *name-independent* – in which nodes have arbitrarily chosen names, *dynamic* – an online version of the problem in which nodes dynamically join and leave the multicast service and the goal is to minimize both the cost of the multicast tree at each stage and the total cost of control messages needed to update the tree.

1 Introduction

As the Internet becomes increasingly used for wide-scale broadcast of information, the ability to send packets to large fractions of the Internet at near-optimal cost may be the vital step that will allow the Internet to replace traditional broadcast media. For large multicast groups, there are substantial inefficiencies that result from using unicast to send messages to many recipients, both from the standpoint of the sender, and from the standpoint of wasted aggregate bandwidth. However, present-day routers cannot possibly incorporate full global information about all possible multicast sets, nor can they store the complete network graph and calculate the minimum spanning tree (MST) or minimum Steiner tree on the set of destinations. Therefore, it is absolutely crucial that memory is efficiently utilized within the routing fabric.

In this paper, we initiate the study of *compact multicast routing*. Informally, the multicast routing problem it to determine the memory held for routing by each network node, and to devise a routing algorithm that delivers a packet with multiple targets to its destinations. A routing scheme is *compact* if it is memory efficient. Its goodness is measured by its *stretch*, the total network distance it utilizes, compared with the shortest multicast path available. Precise problem definition and performance measures are provided below, and are among the

contributions of this paper. Additionally, we provide multicast routing schemes whose memory/stretch tradeoffs are comparable to the unicast case.

There is a very extensive literature on compact unicast routing [10,12]. Suppose we are given a set of n vertices V with distinct labels from $\{1, \ldots, n\}$, and a weighted, undirected graph G on those vertices. A routing scheme is a distributed algorithm that produces a path in G between $u, v \in V$. The *stretch* of the scheme is defined as the maximum, over all choices of $u, v \in V$, of the length of the produced path between u and v divided by the length of the shortest such path. The *memory* of the scheme is the maximum amount of information the algorithm requires to be stored at a node for the node to locally produce the next hop of any routing path. The *header size* of the scheme is the amount of information the scheme requires to be included in each packet to be routed. The challenge is to produce *compact routing schemes* that use $o(n)$ memory and poly-log sized headers, while producing $O(1)$ stretch.

Much is known about these tradeoffs. When nodes are allowed to be renamed, it is known that producing stretch $2k-1$ requires $\widetilde{O}(n^{1/k})$ memory [12], and this matches the lower bound known for $k = 3, 5$. For trees, routing requires $O(\log^2 n / \log \log n)$ memory and header size, and this is known to be optimal [6]. When nodes are given fixed labels, it is known that a stretch 3 scheme exists requiring $\widetilde{O}(n^{1/2})$ memory [2], and a stretch $O(k)$ scheme exists with memory $\widetilde{O}(kn^{1/k})$ [1].

We propose the existence of *compact multicast routing schemes*, which are distributed algorithms to deliver a packet from any node $u \in V$ to any set of target nodes $A \subseteq V$. The *memory* of the scheme is defined as in the unicast problem, but stretch is now defined as the maximum, over all choices of u and A, of the total weight of edges used by the algorithm to deliver the packet to all nodes of A, divided by the weight of the minimum Steiner tree with $A \cup \{u\}$ as targets. We allow headers to be $\widetilde{O}(|A|)$ bits in size, so that the packets can include a list of all destinations, but are similarly restrained, as in the unicast case, from including full path information.

At first sight, the problem seems daunting. There are $2^n - 1$ possible destination subsets. So a complete information scheme should maintain at each node the next step(s) of the shortest spanning tree, for all possible subsets. So it might appear that an optimal solution that locally determines how to forward a packet to each possible target set requires exponential memory. Fortunately, this is not so, as the entire graph can be stored using $\widetilde{O}(n^2)$ memory (more precisely, using $\widetilde{O}(|E|)$ memory), from which a 2-approximation of the optimal solution can be derived in reasonable computation. Therefore, from here on, we refer to any solution that stores $o(|E|)$ as compact multicast routing. We are also interested in the total space consumption of a scheme, and desire to have $o(n \times |E|)$.

Contributions. Our contributions include the following.

1. A formulation of the *compact multicast routing problem* and its performance measures.

2. A labeled multicast routing scheme whose memory requirement at each node is $\widetilde{O}(n^{1/k})$, uses labels of size $\widetilde{O}(n^{1/k})$, employs headers of size $\widetilde{O}(n^{1/k})$, whose stretch is $4k-2$.
3. A name-independent multicast routing scheme whose total memory is $\widetilde{O}(kn^{1+1/k}\log\Delta)$ with stretch $O(k)$.
4. A name-independent multicast routing scheme for growth bounded networks whose memory is $\widetilde{O}(kn^{1/2+1/k})$ per node with stretch $O(k)$.
5. A definition of the *dynamic multicast routing problem*, allowing destinations to be added and removed incrementally to the multicast path.
6. A dynamic multicast scheme that adapts to online node joins with polylogarithmic memory per node and $O(\log^2 n)$ competitive ratio.
7. A dynamic multicast scheme that adapts to online node joins and leaves with polylogarithmic memory per node and $O(\log^3 n)$ competitive ratio.

Related work. Our constructions uses building blocks from compact unicast routing schemes. Among them we use: the labelled tree routing of Thorup & Zwick [12] and Fraigniaud & Gavoille [6], the distance oracles of Thorup and Zwick [12,13], the sparse partitions of Awerbuch and Peleg [4]. We details these results in Section A.

The non-distributed dynamic multicast problem is related to the online steiner tree problem [8,5] in which a similar join-only problem (without leave events) is studied in a centralized model. In contrast our problem concerns a distributed setting and requires nodes to leave the services at a competitive cost. Awerbuch, Bartal and Fiat study the distributed file allocation problem [3]. The join-only dynamic multicast problem can be viewed as a variation of their problem limited to read only requests and serving each request by creating a copy at the reader. In their setting, they achieve a $O(\log^3 n)$ competitive ratio, while our join-only scheme achieves an $O(\log^2 n)$ ratio. Jia et al. [9] propose a single source *universal* scheme in which a *single* tree provides a $O(\log^4 / \log\log n)$ competitive steiner tree for *any* target set. Gupta et al. [7] claim to improve the bound to $O(\log^2 n)$. We use a construction based on [7] in our dynamic join-leave problem. Our scheme implicitly builds a universal tree for *every* target in a non-trivial manner while storing only a polylogarithmic number of bits per node. Moreover, our resulting scheme is *oblivious*, the path from the source to a given target is irrespective of the current set of other targets. We utilize this fact to efficiently handle leave events.

2 Preliminaries

Let $G=(V,E,\omega)$ be an undirected graph with $n=|V|$ nodes and non-negative weight function $\omega : E \to \Re^+$. For an edge set $P \subseteq E$ let $G(P)$ be the subgraph of G induced by the nodes in P. Let the cost of an edge set be the sum of weights of it edges, for a set of nodes $S \subset V$ let $d_G(S)$ be their *Steiner tree* cost, the cost of the minimum cost edge set P such that S is connected in $G(P)$ (this must

be a tree). When G is clear from the context we omit it from the notation and write $d(S)$. When $|S| = 2$, we write $d(u, v)$ as the cost of the minimal cost path between u and v. When $S = V$ then $d(S)$ is the cost of a minimal cost spanning tree on G. Let Δ be the ratio between $\max d(u, v)$ and $\min d(u, v)$ (also called the *aspect ratio* of G).

Consider a subset $A \subseteq V$ of 'target' nodes, and an origin node s. We denote by $S = \{s\} \cup A$ the set containing both origin and target nodes. Let $m = |S|$. Our focus is on the following problems:

- **Labeled compact multicast routing scheme**. This consists of a labeling of nodes and a routing function. The source gets a list of all targets and needs to create a multicast tree.
- **Name-independent compact multicast routing scheme**. This consists of a routing function. The source gets a list of all targets and needs to create a multicast tree. In this model the names of the nodes are chosen arbitrarily as a permutation of $\{1, \ldots, n\}$.
- **Dynamic compact multicast routing scheme**. This consists of a labeling of nodes and a routing function. The source gets an arbitrary sequence join and leave events in an online manner. For each join event it receives the label of the new target and it adds edges to the multicast to connect the target.

We measure the performance of a compact multicast scheme based on the following criteria:

Label size: The maximum number of bits of the label assigned to each node.
Header size: The maximum number of bits sent in a message header.
Total space: The total number of bits used used to store the routing tables.
Max space: The maximum number of bits used at any node to store its routing table.
Stretch: The worst-case ratio, over all sets of destinations, of the sum of the weights of edges used in multicasting to the destinations to the optimal such tree.

In Section 6 we give additional competitive measures for the dynamic online version of the problem.

3 Labeled Compact Multicast Routing

Our first approach builds a multicast route using a Steiner-tree approximation over an approximation of the graph induced by a distance labeling scheme.

Theorem 1. *Let G be a weighted, undirected graph. There is a compact multicast routing scheme for G whose memory requirement at each node is $\widetilde{O}(n^{1/k})$, uses labels of size $\widetilde{O}(n^{1/k})$, employs headers of size $\widetilde{O}(n^{1/k})$, whose stretch is $4k - 2$.*

The scheme makes use of the distance labels and handshake routing scheme of Lemma 9.

Labeling. The labels of nodes and the routing tables are the ones induced by Lemma 9, i.e., node v is specified by its label TZlabel(v).

Storage. Each node v maintains the routing table information TZtable(v) induced by Lemma 9.

Routing. For any two nodes u, v let DO(u, v) be the distance estimation induced by the labels. Let H be a weighted graph on nodes S such that weights between nodes $u, v \in S$ is DO(u, v). Let T be a minimum cost tree on H whose cost is $d_H(S)$. The source computes T in $O(m^2)$ time, and creates a header that contains a *current* node (initially set to s) and the tree T, where each node u of T is designated using its label TZlabel(u).

A node u that receives such a header does the following: If u is the current node of the header then it finds all the edges in T that u needs to send along. For each such edge (TZlabel(u), TZlabel(v)), it creates a header with current node v and forwards it using the handshake routing mechanism and the header information.

Otherwise, if u is not the current node then using the handshake routing scheme, u forwards to the next hop towards the current destination.

Analysis. The storage and header sizes of the scheme are immediate from Lemma 9. We prove the stretch below.

Lemma 1. *The stretch of the multicast routing scheme above is* $4k - 2$

Proof. Let $\bar{H}$ be a weighted graph on S such that weights between nodes $u, v \in S$ is $d(u, v)$. It is well known that minimum cost tree on $\bar{H}$ is a 2 approximation[3] of the cost of the steiner tree of S on G, $d_{\bar{H}}(S) \leq 2d_G(S)$. Hence we only need to show that $d_H(S) \leq (2k-1)d_{\bar{H}}(S)$.

Let $\bar{T}$ be a minimum cost tree on $\bar{H}$ whose cost is $d_{\bar{H}}(S)$. Using the fact that $d_H(u, v) \leq (2k-1)d_{\bar{H}}(u, v)$ for any u, v, we have that the cost of the tree $\bar{T}$ on H is at most $(2k-1)d_{\bar{H}}(S)$. Since T is a minimum cost tree on H then its cost is also at most $(2k-1)d_{\bar{H}}(S)$.

4 Name-Independent Compact Multicast Schemes

The construction in the previous section may require labels of size $\Omega(n^{1/k} \log n)$ per node. In this section, we remove the use of labels. That is, we assume that an origin s is given the set of targets $A = \{t_1, \ldots, t_m\}$ using their original network names (for example, taken out of $\{1, \ldots, n\}$. Hence, this solution variant is called *name-independent*. As a consequence of name-independence, we also avoid the

[3] There exist better approximations (e.g., see [11]), which may shave off a factor of almost two from our scheme. However, our stress in this exposition is on clarity, rather than optimal constants.

need to carry around lengthy labels in packet headers. Headers in our name-independent schemes below are of size proportional to the target list, $\widetilde{O}(m)$.

We remark that a trivial compact name independent multicast scheme can be derived by requiring each node to maintain the distance oracle of Lemma 9. Then employ it to map the set of unlabeled destination into a labeled set, and use the labeled scheme of Section 3 above. The memory requirement at each node is $\widetilde{O}(n^{1+1/k})$. Although this is indeed a compact solution (compared with $\widetilde{O}(|V|+|E|)$ memory per node), the memory at every node is super-linear, which might be prohibitive for large networks. We can bring down the total memory consumption by storing information selectively at key points in the network, and looking it up when needed.

Theorem 2. *Let G be a weighted, undirected graph with aspect ratio Δ. There is a name-independent compact multicast routing scheme for G whose total memory requirement is $\widetilde{O}(kn^{1+1/k}\log\Delta)$, uses the original network labels (of size $\widetilde{O}(1)$), employs headers of size $\widetilde{O}(m)$, whose stretch is $20k-10$.*

The rest of this section is devoted to the proof of Theorem 2. We use the sparse partitions building block given in Theorem 10: Let $I=\{0,1,2,\ldots,\lceil\log\Delta\rceil\}$. The bundle $\mathcal{B}_k$ consists for $i\in I$ of $\mathcal{TC}_{k,2^i}(G)$, a sparse cover of radius 2^i with parameter k.

For every $T\in\mathcal{TC}_{k,2^i}(G)$, denote by $c(T)$ its center node (the center is the seed node from which the cluster was grown). For any node v and index $i\in I$, denote by $T_i(v)$ the tree in $\mathcal{TC}_{k,2^i}(G)$ that contains $B(v,2^i)$.

Storage.

1. Each $v\in V$ stores $\mathrm{TZlabel}(c(T_i(v)))$ for all $i\in I$, where $\mathrm{TZlabel}(\cdot)$ is given by Lemma 9.
2. Each $v\in V$ stores the routing table information $\mathrm{TZtable}(v)$.
3. For each $i\in I$ and $T\in\mathcal{TC}_{k,2^i}(G)$, node $c(T)$ stores the mapping $v\longrightarrow\mathrm{TZlabel}(v)$, from node names to labels, for each node $v\in T$.

Lemma 2. *The total storage used by all nodes is $\widetilde{O}(kn^{1+1/k})$ bits.*

Proof. By Lemma 10, for every $i\in I$, every node v belongs to at most $2kn^{1/k}$ cover-sets of $\mathcal{TC}_{k,2^i}(G)$. According to Lemma 9, a label $\mathrm{label}(v)$ has size $\widetilde{O}(n^{1/k})$. Therefore, the total memory used for label-maps by all $c(T)$, where $T\in\mathcal{TC}_{k,2^i}(G)$ is at most $n\cdot 2kn^{1/k}\cdot O(n^{1/k})=O(kn^{1+1/k})$. The total memory over all $|I|$ radii is $O(kn^{1+1/k}\log\Delta)$. In addition, each node v stores $O(\log\Delta)$ labels of size $\widetilde{O}(kn^{1/k})$ each. The total memory consumption of $\widetilde{O}(kn^{1+1/k})$ bits.

Routing. We need to devise a strategy for an origin s to obtain the labels of the multicast targets without going too far. This is achieved as follows.

Observe that $c = c(T_i(s))$ is at distance at most $k2^i$ from s, and c stores label mappings for all $v \in B(s, 2^i)$. Therefore, with cost $O(k2^i)$, s can send S to $c(T_i(s))$, and let $c(T_i(s))$ compute the required multicast route. By repeatedly trying for increasing distances 2^i, s can find the appropriate $c(T_i(s))$ at a competitive cost.

The multicast strategy is as follows. Node s iteratively sends a request for help containing the target list S to nodes $c(T_i(s))$ for $i \in I$, until it reaches a node $c = c(T_i(s))$ such that c stores the labels of every node in S. Then node c computes the header as in Theorem 1.

Lemma 3. *The stretch of the multicast routing scheme is* $20k - 10$.

Proof. Let j be the index such that $2^{j-1} < \max_{a \in A} d(s, a) \leq 2^j$. Then clearly $S \subseteq T_j(s)$ and $2^j < 2d(S)$. Since the radius of a level 2^i cover is $(2k-1)2^i$ then the cost of going from s to $T_i(s)$ and back is at most $2(2k-1)2^i$. And the total cost to reach $T_j(s)$ is

$$\sum_{1 \leq i \leq j} 2(2k-1)2^i \leq 4(2k-1)2^j \leq 8(2k-1)d(S)$$

In addition, the cost of the multicast scheme of Theorem 1 is at most $(4k-2)d(S)$.

5 Balanced Name Independent Compact Multicast Routing for Growth Bounded Networks

In this section we provide a balanced scheme for growth bounded networks. Formally, for a node $v \in V$ and $i \in \mathbb{N}$ let $N_v(i)$ denote the i closest nodes to v with ties broken by lexicographical order.

We assume the network is δ-*growth bounded*, for $\delta \geq 2$, such that

$$\operatorname{diam}(N_v(2i)) \geq \delta \cdot \operatorname{diam}(N_v(i)) \ .$$

The Single Source Directory Scheme (SSD)

We begin by building a single source directory (SSD) scheme. For a network of n nodes and a parameter k, the scheme requires each node to store $\widetilde{O}(n^{1/2+1/k})$ bit of storage. The scheme allows a fixed source c to find all the labels of a set A of target nodes with cost that is proportional to the farthest node in A from c. The result is stated in the following lemma:

Lemma 4 (Single-Source Directory (SSD)). *Let $F = (V, E, \omega)$ be a weighted graph, $|V| = n$, $c \in V$ a given source node. There exists a multi-node lookup scheme as follows.*

- *Given a set of target nodes $A \subseteq V$, the source c can find the labels* $\mathrm{TZlabel}(a)$ *of all nodes $a \in A$, where* $\mathrm{TZlabel}(a)$ *is determined as in Lemma 9.*
- *The scheme requires each node to store $\widetilde{O}(kn^{1/2+1/k})$ bits.*
- *The length of the path used for finding the labels in A is at most* $4 \max_{a \in A} d(c, a)$.

SSD Storage. Let c be the source node. Let $\Gamma = \sqrt{n}$. Enumerate the nodes in $N_c(\Gamma)$ arbitrarily $w_1, w_2, \ldots, w_\Gamma$. For any integer $i < \log \Gamma$ let the nodes $w_{2^i}, w_{2^i+1}, \ldots, w_{2^{i+1}-1}$ store the labels of all the nodes in $(N_c(\Gamma 2^{(i+1)}) \setminus N_c(\Gamma 2^i)$. Note that each w_j stores $\sqrt{n}$ labels. Node c itself stores the labels of $N_c(\Gamma 2^0)$, i.e., of the $\sqrt{n}$ nodes closest to it.

SSD Lookup. It is left to show how to obtain the labels of targets with a competitive cost. This is done as follows.

1. If $A \subseteq N_c(\Gamma)$, then c has all the labels already.
2. Otherwise, set $i = 0$, and set $A_0 = A \setminus (A \cap N_c(\Gamma))$. That is, A_0 contains the remaining targets for which c does not have labels yet. Repeat for $i = 0, 1, 2, \ldots,$ until A_i is empty:
   - Node c queries from $w_{2^i}, w_{2^i+1}, \ldots, w_{2^{i+1}-1}$ the labels of any target nodes in A_i. Note that in response, c should obtain the labels of all nodes in $A_i \cap N_c(\Gamma 2^{i+1})$.
   - Then set $A_{i+1} = A_i \setminus N_c(\Gamma 2^{i+1})$, set $i = i + 1$, and repeat.

SSD Analysis.

Proof (Proof of Lemma 4). let $d = \mathrm{diam}(N_c(\Gamma))$. An invariant maintained in the algorithm is as follows. At step i, there is a target $a \in (A \setminus N_c(\Gamma 2^i)$. Hence, if step i is reached, there is a target $a \in A_i$ whose distance from c is at least $\mathrm{diam}(N_c(\Gamma 2^i))$. By the growth bound,

$$\mathrm{diam}(N_c(\Gamma 2^i)) \geq 2^i d \ .$$

Therefore, $\max_{a \in A} d(c, a) \geq 2^i d$. The total cost of steps $j = 1, \ldots, i$ is a geometric series, whose sum is bounded by $\sum_{j=0..i} 2 \times 2^j \leq 4 \times 2^i$. This yields that the total distance of lookups is at most $4 \max_{a \in A} d(c, a)$.

The Full Scheme

As in the un-balanced name-independent multicast scheme of Section 4, we make use of sparse partitions as given in Theorem 10.

Storage.

1. Each $v \in V$ stores $\mathrm{TZlabel}(c(T_i(v)))$ for all $i \in I$.
2. Each $v \in V$ stores the routing table information $\mathrm{TZtable}(v)$.
3. For each $i \in I$ and $T \in \mathcal{TC}_{k,2^i}(G)$, employ the scheme of Lemma 4 with T the graph, and $c(T)$ the source node. (Recall that this allows $c(T)$ to look up the labels $\mathrm{TZlabel}(a)$ of any set of targets A, such that $a \in A$, with cost at most $4\mathrm{diam}(T)$.

Lemma 5. *The total storage used by any node is* $\widetilde{O}(kn^{1/2+1/k})$.

Routing. Given a set of targets A, s computes a Steiner graph spanning A in two steps: First, it obtains the labels of all the nodes in A; then it computes an approximate Steiner graph as in the labeled scheme of Section 3.

We need to devise a strategy for an origin s to obtain the labels of the multicast targets without going too far. This is achieved as follows.

Observe that $c = c(T_i(s))$ is at distance at most $(2k-1)2^i$ from s. By Lemma 4, from origin c a lookup of label mappings for all $v \in B(s, 2^i)$ has a cost bounded by $4 \times (2k-1)2^i$. Therefore, with cost $O(k2^i)$, s can send A to $c(T_i(s))$, and let $c(T_i(s))$ return the required labels. By repeatedly trying for increasing distances 2^i, s can find the appropriate $c(T_i(s))$ at a competitive cost.

The multicast strategy is as follows. Node s iteratively sends a request for help containing the target list A to nodes $c(T_i(s))$ for $i \in I$, until it reaches a node $c = c(T_i(s))$ such that c can find the labels of every node in A. Then node c computes the header as in Theorem 1.

Lemma 6. *The stretch of the multicast routing scheme is* $36k - 18$.

Proof. Let j be the index such that $2^{j-1} < \max_{a \in A} d(s,a) \leq 2^j$. Then clearly $S \subseteq T_j(s)$ and $2^j < 2d(S)$. For every $i \leq j$, since the radius of a level 2^i cover is $(2k-1)2^i$ then the cost of going from s to $c(T_i(s))$ and back is at most $2(2k-1)2^i$. The cost of searching for label mappings in $T_i(s)$ is bounded according to Lemma 4 by $2(2k-1)2^i$. Hence, the total cost to collect all required label mappings is:

$$\sum_{1 \leq i \leq j} 4(2k-1)2^i \leq 8(2k-1)2^j \leq 16(2k-1)d(S)$$

Finally, the cost of the Steiner tree computed using these labels is bounded according to Lemma 1 by an additional $(4k-2)d(S)$.

6 Dynamic Compact Multicast Routing Schemes

A weakness of all the solutions above is that the multicast dissemination path must be re-computed each time a target wants to join or eave the multicast service. In a highly dynamic setting in which the multicast set is constantly changing rebuilding the multicast tree from scratch after each change may be unacceptable.

Consider a source node s and a sequence of node join and node leave events. In such a dynamic setting there are two measures one would like to minimize. The first is the *communication-cost*, namely, the total cost of the edges used during the algorithm while building the multicast trees of the various stages. The second is the *multicast-cost*, which for each stage is the cost of the current multicast tree.

We now explain why communication-cost alone does not suffice to bound the cost of a multicast scheme. In order to understand this, consider a simple-path

graph $s = v_1 - v_2 - ... - v_n$, and suppose that the sequence of joins/leaves is the following: join(v_n), join(v_{n-1}), ..., join(v_2), join(v_1), leave(v_n), leave(v_{n-1}), ..., leave(v_2). If we consider only the total communication cost, we need not ever remove edges from the multicast path, which is the whole graph in this case. In the end, we may be left with a very inefficient multicast graph containing v_1 and v_2, but the communication cost measure does not capture this. Therefore, we must also consider the efficiency of the current multicast path at every step, which is precisely captured by multicast-cost.

In order to formally bound theses measures, we make use of *online analysis*. Specifically we compare both costs to an optimal off-line algorithm. The off-line algorithm knows the sequence of joins/leaves in advance, and has no extraneous communication cost in setting up the multicast overlay. The communication-cost for the online algorithm is therefore the total cost of edges used for the optimal Steiner trees at different steps.

An on-line algorithm is (α, β) competitive for the dynamic multicast problem if for any source and for any sequence of join/leave events the total cost of the set of edges used is at most α times that of the total cost of the set of edges used by the optimal algorithm and at *each* stage the cost of the current multicast tree is at most β times that of the cost of the current optimal algorithm.

Let A be the set of all nodes that joined during a sequence. Then any algorithm must pay at least $d(A \cup \{s\})$ for communication-cost, and if A' is the current set of nodes of a given stage then any algorithm must pay $d(A' \cup \{s\})$ multicast-cost for this stage.

In this extended abstract we handle the labeled variant of this problem in which nodes are labeled and the sequence of join/leave events are given in an online manner to the source. In the full paper we will show how to extend these ideas for a name-independent variant of the problem.

We begin by presenting our first scheme that only allows join events to the multicast service over time (but no leave events). We prove it is $(O(\log^2 n), O(\log^2 n))$ competitive. This may be appropriate in settings where once a node joins a multicast service it will not leave until the multicast is complete. Our second scheme handles both join and leave events at a cost of a being $(O(\log^3), O(\log^3))$ competitive.

6.1 Dynamic compact multicast routing schemes for join-only events

Assume the sequence of node joins is $A = \{a_1, a_2, \dots\}$. The scheme employs a bundle $\mathcal{B}_k = \{\mathcal{TC}_{k,2^i}(G) \mid i \in I\}$ of Sparse Covers of Lemma 10 with $k = \log n$. For a node v, denote by $\mathcal{B}(v)$ the set of all covers T in the bundle $\mathcal{B}_k$ such that $v \in T$.

Labelling. The label SPlabel(v) stores the label $\lambda(T, c(T))$ given by Lemma 9 for each $T \in \mathcal{B}(v)$. Note that the label size is $O(\log^3 n \log \Delta / \log \log n)$.

Storage. Each node v stores tree routing information $\mu(T, v)$ for all the trees in its own label SPlabel(v) (recall that SPlabel(v) consists of $\lambda(T, c(T))$ for every tree-cover $T \in \mathcal{B}$ containing v). The total storage is $O(\log^3 n \log \Delta / \log\log n)$.

Multicast-graph Construction. The construction of the multicast graph is done in steps. At step j, node a_j is brought into the multicast graph. This entails informing all relevant nodes in the graph who should become their new neighbors.

For convenience, we denote by $A_j = \{s, a_1, \ldots, a_j\}$. The key is to maintain for each $i \in I$ a set $U_i \subset A_j$ such that if $u, v \in U_i$ then $d(u, v) \geq 2^i$ and for any $v \in A_j$ there exists $u \in U_i$ such that $T_i(u) \in \mathcal{B}(v)$.

Initially $U_i = \{s\}$ for all $i \in I$. Given a new node a with label SPlabel(a), let i^* be the maximal index i such that $\mathcal{B}(a) \cap \{T_i(u) \mid u \in U_i\} = \emptyset$ then we connect a to the existing multicast tree by a route from a to $c(T_{i^*+1}(u))$ and from $c(T_{i^*+1}(u))$ to u where $u \in U_{i^*+1}$ and $T_{i^*+1}(u) \in \mathcal{B}(a)$. We update the sets by adding a to U_i for all $i \in \{0, 1, 2, \ldots, i^*\}$. Note that it suffices for s to store the tuple $(u, c(T_{i^*+1}(u)), \text{TZlabel}(T_{i^*+1}(u), a))$ for the multicast route to be able to reach a.

Analysis. It is immediate to see that for each $i \in I$ the set U_i maintains the property that for any $v \in A_j$ there exists $u \in U_i$ such that $T_i(u) \in \mathcal{B}(v)$. To see that if $u, v \in U_i$ then $d(u, v) \geq 2^i$ note that if $d(a, u) < 2^i$ for some $u \in U_i$ then it must be that $a \in T_i(u)$ so $T_i(u) \in \mathcal{B}(a)$ and thus a will not be added to U_i.

Since the model allows only join events the communication cost and the multicast cost are both bounded by the cost of the Steiner tree on the set of current nodes.

Lemma 7. *The dynamic multicast algorithm for joins is $O(\min\{\log n, \log \Delta\} \log n)$ competitive both in communication cost and in multicast cost.*

Proof. In the case of joins only, the communication cost and multicast cost of the off-line algorithm are the same, namely, the cost of optimal Steiner tree of all targets. For each $i \in I$ for which $|U_i| > 1$ any algorithm must pay at least $|U_i| 2^{i-1}$ since the balls of radius 2^{i-1} around members of U_i are disjoint. On the other hand our algorithm pays at most $|U_i| 2^i \cdot 2 \log n \cdot 2$ for connecting nodes in U_i.

Let d be the diameter of A then for each $i \leq \log d \in I$ we pay at most $4 \log n$ times the optimal. If $\log \Delta$ is large we note that it is actually enough to look at the $4 \log n$ levels $\{\log d, \log d - 1, \ldots, \log d - 4 \log n\}$, as the cost of all the edges of the lower levels will add only a constant factor to the overall cost.

6.2 Fully dynamic compact multicast routing scheme

Consider a finite sequence of node join and node leave events. For each stage j let A_j be the current set of multicast targets. Note that at each stage either one node joins or one node leaves, so $|A_{j+1} \ominus A_j| = 1$ (where $X \ominus Y$ is the set of all

nodes x such that either $x \in X$ or $x \in Y$ but not both). Let A be the set of all nodes that joined the multicast service.

Recall the following definition from Gupta et al. [7].

Definition 1 (α-padded nodes). *Given a hierarchical decomposition $P = (P_i)_{i=0}^{h}$, a node $v \in V$ is α-padded in P if for all $i \in [0,h]$, the ball $B(v, \alpha 2^i)$ is contained in some cluster of P_i.*

We use the Theorem 2.2 of Gupta et al. [7] to get a set of $O(\log n)$ hierarchical decompositions $\mathcal{P}$ such that for node u there exists $P \in \mathcal{P}$ such that u is $\Omega(1/\log n)$-padded in P. We denote the decomposition by $P^{(u)}$. If there exists several such $P \in \mathcal{P}$ then define $P^{(u)}$ as the lexicographically first.

Recall that a hierarchical decomposition P induces a dominating tree $T(P)$ simply by associating each cluster with the center node of the cluster and the edge between a cluster and its parent cluster is simply the shortest path between the two cluster's centers. Routing on such a dominating tree can be done using Lemma 8 for deciding which "tree-edge" to take. Each tree edge between a node $u \in T$ that corresponds to a cluster in P_i and a node $v \in T$ that corresponds to a cluster in P_{i+1} induces a path in the graph that is part of a shortest path tree emanating from v and spanning all the nodes corresponding to the cluster in P_{i+1}. Hence we use Lemma 8 again for routing on the shortest path in G from u to v.

A major property of the construction of the multicast graph is that it is *oblivious*. The path from the source to a given target is irrespective of the current set of other targets.

The path of a target a is simply the path induced on G by the path on the tree $T(P^{(a)})$ from a to the source. Note that it is possible to route along this path using Lemma 8 while routing on a shortest path from each node in the sequence to its neighbor in the sequence.

Due to obliviousness, when other nodes join and leave the multicast service this does not effect the path taken to a. When a leaves the source simply stops sending along the edges that are not required any more. This can be implemented by maintaining a simple reference counter on the edges.

Analysis. For any hierarchical decomposition P and scale i, let $X_{P,i}$ be a $2^{i-6}/\log n$-net of the nodes $(A \cup \{s\}) \cap P_i$ (an r-net is a maximal set of nodes whose distance from each other is at least r). For any P and i, even the optimal offline solution must pay $|X_{P_i}|2^{i-7}/\log n$. We will now show that the total cost of edges between clusters of P_{i+1} to clusters of P_{i+2} is at most $|X_{P,i}|2^{i+3}$. Summing over all $\log \Delta$ scales and all $O(\log n)$ trees (each tree is induced by a hierarchical decomposition $P \in \mathcal{P}$) gives a competitive ratio of $O(\log^2 n \min\{\log \Delta, \log n\})$. To see that the total cost of edges between the centers of clusters of P_{i+1} to centers of clusters of P_{i+2} is at most $|X_{P,i}|2^{i+3}$, consider any $a \in A$ that uses P to reach the souse (hence $P^{(a)} = P$). By definition there exists $x \in X_{P,i}$ such that $d(x,a) \leq 2^{i-6}/\log n$. Let $C(x)$ be the cluster in P_{i+1} that contains x. Since

x is fully padded then $a \in C(x)$. Hence the path from a to s and the path from x to s both meet at the center of $C(x)$ so the total cost of the path from the center of $C(x)$ to its parent cluster center in P_{i+2} for all $a \in C(x)$ is at most 2^{i+3}.

7 Conclusions

We have initiated the study of distributed compact multicast routing schemes. No lower bounds are known (other than the known ones for unicast routing). It is an interesting open question to find the optimal trade-offs between storage and space for the various problems considered in this paper.

References

1. Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Routing with improved communication-space trade-off. In 18^{th} *International Symposium on Distributed Computing (DISC)*, volume 3274 of Lecture Notes in Computer Science, pages 305–319. Springer, October 2004.
2. Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 20–24. ACM Press, 2004.
3. Baruch Awerbuch, Yair Bartal, and Amos Fiat. Distributed paging for general networks. In *SODA '96: Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 574–583, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.
4. Baruch Awerbuch and David Peleg. Sparse partitions. In 31^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 503–513. IEEE Computer Society Press, October 1990.
5. Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 344–353, New York, NY, USA, 1997. ACM Press.
6. Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of Lecture Notes in Computer Science, pages 757–772. Springer, July 2001.
7. Anupam Gupta, Mohammad T. Hajiaghayi, and Harald Räcke. Oblivious network design. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 970–979, New York, NY, USA, 2006. ACM.
8. M. Imase and B.M. Waxman. Dynamic steiner tree problem. *SIAM J. Disc Math.*, 4:369–384, 1991.
9. Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for tsp, steiner tree, and set cover. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 386–395, New York, NY, USA, 2005. ACM Press.

10. David Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM Monographs on Discrete Mathematics and Applications, 2000.
11. Gabriel Robins and Alexander Zelikovsky. Improved steiner tree approximation in graphs. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 770–779, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.
12. Mikkel Thorup and Uri Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, Hersonissos, Crete, Greece, July 2001. ACM PRESS.
13. Mikkel Thorup and Uri Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. ACM Press, July 2001.

A Building Blocks

Our schemes make use of several graph partitioning and routing techniques. For clarity, we group them in this section.

Labelled tree routing of Thorup & Zwick [12] and Fraigniaud & Gavoille [6]:

Lemma 8. [6,12] *For every weighted tree T with n nodes there exists a labeled routing scheme that, given any destination label, routes optimally on T from any source to the destination. The storage per node in T, the label size, and the header size are $O(\log^2 n/\log\log n)$ bits. Given the information of a node and the label of the destination, routing decisions take constant time.*

For a tree T containing a node v, we let $\mu(T,v)$ denote the routing information of node v and $\lambda(T,v)$ denote the destination label of v in T as required from Lemma 8.

Distance oracles of Thorup and Zwick [12,13]:

We use their distance labels, handshake routing scheme and distance oracles. The following are simple variations of their results.

Lemma 9 ([12](3.4),(3,1), [13](4.1)). *Let G be a weighted graph with aspect ratio Δ. Let $1 \leq k \leq \log n$ be an integer.*

1. *It is possible to assign to each point $v \in V$ an $O(n^{1/k}\log^{1-1/k} n \log(n\Delta))$-bit label, denoted* $\mathrm{TZlabel}(v)$*, such that given* $\mathrm{TZlabel}(u)$ *and* $\mathrm{TZlabel}(v)$*, for any two points $u, v \in V$, it is possible to compute, in $O(k)$ time, an approximation to the distance $d(u,v)$ with stretch of at most $2k-1$.*
2. *It is possible to assign each node a $\widetilde{O}(kn^{1/k})$ bit routing table, denoted* $\mathrm{TZtable}(v)$*, such that given* $\mathrm{TZlabel}(v)$ *any source u can extract $o(\log^2)$ bits and use them as a header to route from the source to u with cost that equals the cost of the distance estimation obtained by* $\mathrm{TZlabel}(u)$ *and* $\mathrm{TZlabel}(v)$ *via item (1).*

3. *It is possible to create a data structure with size* $O(kn^{1/k}\log(n\Delta))$ *bits, such that distance queries can be answered with the same cost of the distance estimation of item (1). Given* TZlabel(u) *and* TZlabel(v) *it is possible to extract* $o(\log^2 n)$ *bits such that routing from source to target will also have the same cost.*

Sparse partitions of Awerbuch and Peleg [4]:

Lemma 10. [4] *For every weighted graph* $G = (V, E, \omega)$, $|V| = n$ *and integers* $k, \rho \geq 1$, *there exists a polynomial algorithm that constructs a collection of rooted trees* $\mathcal{TC}_{k,\rho}(G)$ *such that:*

1. *(Cover) For all* $v \in V$, *there exists* $T \in \mathcal{TC}_{k,\rho}(G)$ *such that* $B(v,\rho) \subseteq T$.
2. *(Sparse) For all* $v \in V$, $|\{T \in \mathcal{TC}_{k,\rho}(G) \mid v \in T\}| \leq 2kn^{1/k}$.
3. *(Small radius) For all* $T \in \mathcal{TC}_{k,\rho}(G)$, *there is a root node* $r \in T$ *such that* $\mathrm{rad}(T) \leq (2k-1)\rho$, *where* $\mathrm{rad}(T) = \max_u\{d_T(r,u)\}$.

Denote by $I = \{0, 1, \ldots, \lceil \log \Delta(G) \rceil\}$. Usually, we make use of a *bundle* $\mathcal{B}_k = \{\mathcal{TC}_{k,2^i}(G) \mid i \in I\}$ of covers and in some cases we make use of a bundle $\mathcal{B}_k^\sigma = \{\mathcal{TC}_{k,\sigma^i}(G) \mid i \in \{0, 1, \ldots, \log_\sigma \Delta(G)\}\}$.

For every $T \in \mathcal{TC}_{k,2^i}(G)$, denote by $c(T)$ its center node (the center is the seed node from which the cluster was grown). For any node v and index $i \in I$, denote by $T_i(v)$ the tree in $\mathcal{TC}_{k,2^i}(G)$ that contains $B(v, 2^i)$.