Topology Aggregation for Directed Graphs

Baruch Awerbuch and Yuval Shavitt

Abstract— **This paper addresses the problem of aggregating the topology of a sub-network in a compact way with minimum distortion. The problem arises from networks that have a hierarchical structure, where each sub-network must advertise the cost of routing between each pair of its border nodes. The straight-forward solution of advertising the exact cost for each pair has a quadratic cost which is not practical. We look at the realistic scenario of networks where all links are bidirectional, but their cost (or distance) in the opposite directions might differ significantly.**

The paper presents a solution with distortion that is bounded by the logarithm of the number of border nodes and the square-root of the asymmetry in the cost of a link. This is the first time that a theoretical bound is given to an undirected graph. We show how to apply our solution to PNNI, and suggest some other heuristics that are tested to perform better than the provenly bounded solution.

I. Introduction

As networks grow in size, the collection and maintenance of control information from the entire network, e.g., to make routing decisions, becomes difficult if not impossible. For large networks, the current solution is to partition the network into subdomains, and continue partitioning the subdomains recursively until the lower level where the subdomain is comprised of subnetworks [PNN96], [CCS96]. At each level, control information is aggregated before shared with peer subdomains. Specifically, for routing purposes, the internal network structure is hidden, and instead, the network presents to the outside world compressed information that enables routing entities to make intelligent decisions as to which subnetwork to use and to select the appropriate entry points to each subnetwork. The mathematical property of routing in such a scenario were studied by Guérin and Orda [GO99].

Methods for compact graph representation that can be used to aggregate topology information were studied in the past years. A spanner [PU88], [PS89], [ADD+93] of a graph G is a subgraph G' that includes all the node of G but only a subset of the edges. Spanners with stretch t, t-spanners, have the property that the cost of the minimum cost path in G' between any two nodes is, at most, a factor t greater than the cost of the minimum cost path between the same nodes in G. We term this factor the maximum distortion of the graph compression, or simply the *distortion*. The best known algorithm [ADD+93] for spanners with stretch $2t+1$ requires $O(b^{1+1/t})$ edges, where b is the number of border nodes, i.e., if a logarithmic distortion is acceptable the spanner size is a constant factor of the number of nodes.

B. Awerbuch is with the Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA. baruch@cs.jhu.edu

Y. Shavitt is with the Department of Electrical Engineering - Systems, Tel-Aviv University, Tel-Aviv 69978, Israel, and Bell Laboratories, Lucent Technologies, 101 Crawfords Corner Rd., Holmdel, NJ 07733, USA. shavitt@ieee.org. This research was done in part while he was with Johns Hopkins University, Baltimore, MD, USA.

Bartal [Bar96] suggests a tree representation for graphs with an average distortion of $O(\log n)$, where n is the number of nodes in the graph. The tree is not built from the edges of the original graph, and is using additional virtual nodes.

The assumption in all the above mentioned aggregations is that the graph metric is additive, i.e., the cost of a path is the sum of the costs of the links comprising it. This is the case where the link cost represents the delay to traverse a link, or a price associated with a link traversal. If the graph metric is calculated with the minimum or maximum function, e.g., maximum available bandwidth, then a tree representation is sufficient for an accurate representation [Lee95]. This work addresses only graphs with additive costs.

All the above aggregation schemes work on undirected graphs. However, in todays application and standards [PNN96], reservations for each direction of the connection may take different values. As a result, the network graph becomes directional, with edges that have different weights for the two directions. For routing, the edge weight is usually a function of the residual bandwidth or the average delay on the link, and these, in case of asymmetric traffic, might have orders of magnitude differences.

For general digraphs, Peleg and Schäffer [PS89] show that for any t, a t-spanner might require $\Omega(n^2)$ links. In other words, for certain directed graphs, no compact representation with bounded distortion exists. However, in practice, the ratio between the weights of a link is bounded by some asymmetry constant, ρ, that depends on the link weight function. For example, a link delay function may be bounded from above due to finite buffer space, and from below due to processing delay. Another reason for a bound on the asymmetry is the usages of fixed size fields to represent link weights.

In this paper, we show a compact representation for a directed graph that are known to have some bound on their asymmetry constant. Our compression distortion depends on the actual asymmetry constant of the graph and not on the bound for this constant. In general, we show that a subgraph of a network with b border nodes can be represented by $O(b)$ weights with $O(\sqrt{\rho}\log b)$ distortion. Note, that in practical cases, ρ is expected to be larger than b by several orders of magnitude.

The fact that link costs can vary over orders of magnitude might look surprising. However, both queueing theory and opportunity cost functions suggest such a wide range. If link cost is to be determined by the expected queueing delay of a packet through this link, which is the major factor in the total delay, the delay function is given by $f \cdot [\mu(c-f)]^{-1}$, where c is the link capacity, f is the current flow through the link, and μ is the service rate.

As f approaches c, the link cost increases to infinity. Opportunity cost functions suggest to maximize utilization by charging link usage according to a function that increases exponentially as the residual bandwidth decreases.

The rest of the paper is organized as follows. In the next section, we describe the network model, and give our notations. In section III we describe the aggregation algorithm and analyze its performance, and in section IV we simulate its performance. Finally we conclude by showing the applicability of our results to the PNNI standard, and suggesting some alternative heuristics that are tested to perform well.

II. Model and Notation

We use the PNNI [PNN96] notation. A subnetwork comprised of V, a group of connected nodes, that is called a peer group. $B \subset V$ is a set of border nodes, i.e., nodes that have a direct link to a node outside of the peer group. Each directed edge $(v,\ u) \in E$ is associated with a weight, $w(v,\ u) \in \mathbb{R}^+$, s.t., $\rho^{-1} \leq w(v,\ u)/w(u,\ v) \leq \rho$. W.l.o.g. let $w(v,\ u) > w(u,\ v)$, then $\rho_{u,v} \equiv \rho_{v,u} \triangleq w(v,\ u)/w(u,\ v) \leq \rho$ and $\rho = \max_{(u,v):u,v\in V} \rho_{u,v}$. ρ is the network asymmetry constant, $\rho_{u,v}$ is the asymmetry factor for a node pair.

Since the internal structure of the graph is irrelevant to route through it, as only the cost matrix is of relevance, the original graph, $G(V,\ E)$, is transformed to a directed clique, $\vec{K}(B)$, where B is the the group of border nodes. Each $(v,\ u)$, a directed link in $\vec{K}(B)$, is associated with a weight, $w(v,\ u) \in \mathbb{R}^+$, that is equal to the weight of the shortest path between v and u in $G(V,\ E)$. By $w(v,\ u)$ definition for a path, the inequality $\rho^{-1} \leq w(v,\ u)/w(u,\ v) \leq \rho$ holds for paths, as well. A full representation of the weight distances (or costs) between the border nodes requires $b(b-1) \in O(b^2)$ space, where b is the number of border nodes.

Since $\vec{K}(B)$ is constructed by assigning each link the cost of the minimum cost path in the original graph, we get the following property. For every $u,\ v,\ x \in B$:

$$w(u,\ v) \leq w(u,\ x) + w(x,\ v)$$

This *directed triangle inequality* suggests that although link weights can be arbitrary different in the original graph, in the resulting clique, link weights are bounded by the weights of their adjacent links.

We transform the directed clique, $\vec{K}(B)$, to an undirected clique, $K(B)$, and later delete links from $K(B)$ to receive the graph $K^-(B)$. $w_K(u,\ v)$ denotes the cost of link $(v,\ u)$ in $K(B)$. $w_{K^-}(u,\ v)$ is the cost of the shortest path between nodes u and v in $K^-(B)$.

III. Algorithm Description and Analysis

Our suggestion for treating asymmetry is simple, yet, powerful. In the first step, the original digraph, $G(V,\ E)$, is transformed to a directed clique, $\vec{K}(B)$. A directed link between node b_i and node b_j in the clique has the weight of the shortest path between these two border nodes in G. Note that this transformation preserves all the routing information, and if the $b(b-1)$ distances between all the node pairs of the clique are broadcast full accurate routing information is available.

We transform the directed clique, $\vec{K}(B)$, to an undirected clique, $K(B)$, as follows. Every pair of directed links $(v,\ u)$ and $(u,\ v)$ with weights $w(v,\ u)$ and $w(u,\ v)$, respectively, is replaced with an undirected link $(v,\ u)$ with weight $w_K(v,\ u) = \sqrt{w(v,\ u) \cdot w(u,\ v)}$. At this stage we can use spanners with logarithmic distortion [ADD+93] and the result requires only $O(b)$ edges. Specifically, a $(2\log_2 b + 1)$-spanner with less than $2b$ edges can be found for $K(B)$ by a simple polynomial algorithm[1]. We note that the distortion for the border-node pair u and v depends only on the number of border nodes in the graph and the graph asymmetry constant, ρ. The overall weight distortion for an edge $(u,\ v)$ can be in the range $[c_l w(u\ ,v)/\sqrt{\rho},\ c_u w(u\ ,v) \cdot \sqrt{\rho_{u,v}} \cdot \log b]$, where c_l and c_u are some constants. The distortion calculation will be explained later after the discussion of the distortion of the tree construction. Remember, that in practical cases, ρ is expected to be larger than b by several orders of magnitude.

Using Bartal's tree construction [Bar96] might be more appealing due to the fact that the resulted aggregated structure has a known structure (a tree), while the distortion remains logarithmic in b. However, Bartal's algorithm cannot be applied directly to the undirected clique since it requires the graph to represent a metric space while the resulted clique may contain triangles that do not obey the triangle inequality as can be shown by the simple three-node example-network in figure 1. In this example, a three-node directed clique with $\rho = 10$ (Fig. 1 above) is transformed to an undirected clique (Fig. 1 below). The resulted triangle does not obey the triangle inequality as $\sqrt{10} + 10 = 13.16 < 10\sqrt{2} = 14.14$

However, a clique, $K(B)$, can be transformed to represent a metric space in the following way. List all the triangles that do not obey the triangle inequality and delete the longest edge in all of them. The resulted graph, $K^-(B)$, represents a metric space on which we apply Bartal's algorithm.

In the following, we investigate the influence the edge deletion has on the distortion. We show first that the graph remains connected, and then we give a bound on the additional distortion due to edge deletion.

Theorem 1: The link deletion from the undirected clique results in a connected graph.

Proof: Assume that the graph $K^-(B)$ is not connected. Thus, two border nodes, u and v, exist in $K^-(B)$ such that no path connects them in $K^-(B)$. Consider a link $(w,\ x)$ on the shortest path between u and v in $K(B)$. To be deleted, $(w,\ x)$ should be longer than the sum of the other two links in a triangle, which contradicts the fact that $(w,\ x)$ belongs to the shortest path. □

[1] The algorithm examines the links in increasing length order, and adds a link only if it shortens the distance between its endpoint by a factor of t, or more. Its running time is thus $|E|$ times running a shortest path algorithm between two nodes [ADD+93].

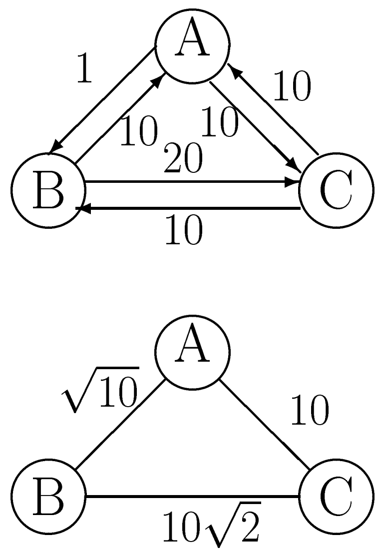

Fig. 1. An example where the transformation from a directed to an undirected clique results in a triangle that does not obey the triangle inequality

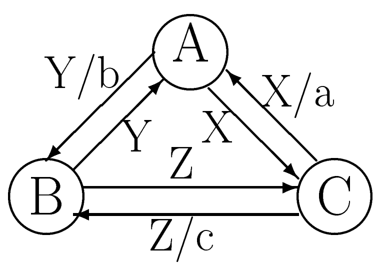

Fig. 2. A general triangle

Lemma 1: For a link with asymmetry ratio ρ_e, the maximum distortion due to link deletion from the undirected clique is $\sqrt{\rho/\rho_e}$.

Proof: Consider the general triangle in figure 2. Let

$$X' = X/a \quad (1)$$
$$Y' = Y/b \quad (2)$$
$$Z' = Z/c \quad (3)$$

where $a,\ b \geq 1$. By the directed triangle inequality

$$Z \leq X + Y \quad (4)$$
$$Z' \leq X' + Y' \quad (5)$$

To maximize the distortion let $Z = X + Y$ which forces $c \geq 1$. W.l.o.g. assume $X \geq Y$. We can write

$$X = pZ \quad (6)$$
$$Y = (1-p)Z \quad (7)$$

where $0.5 \leq p < 1$.

The additional distortion due to the deletion of link (B, C) from $K(B)$ is given by

$$dist = \frac{\sqrt{ZZ'}}{\sqrt{XX'} + \sqrt{YY'}} = \frac{\sqrt{ab/c}}{\sqrt{a} - p(\sqrt{a} - \sqrt{b})} \quad (8)$$

where the second equation in (8) is due to substituting (1) – (7) and performing some simple algebraic manipulations.

If $a > b$ the maximum is achieved when p approaches 1, and the maximum distortion is then

$$dist^* = \sqrt{\frac{a}{c}} = \sqrt{\frac{\max\{a,\ b\}}{c}} \quad (9)$$

If $a < b$ the maximum is achieved when $p = 0.5$ and its value is $\sqrt{\frac{\min\{a,\ b\}}{c}} < dist^*$.

$\sqrt{\frac{a}{c}}$ is the maximum distortion for one triangle. When several triangles are cascaded, let the asymmetry ratio of the longest link be a_0, let the asymmetry ratio of the link sharing a triangle with the longest link be a_1, and define a_i, $i = 2,\ 3,\ ,\ldots, m$ in the same manner (see figure 3). The total distortion due to the deletion of the m links that violates the triangle inequality is

$$\sqrt{\frac{a_1}{a_0}} \cdot \sqrt{\frac{a_2}{a_1}} \cdots \sqrt{\frac{a_m}{a_{m-1}}} = \sqrt{\frac{a_m}{a_0}} \leq \frac{\sqrt{\rho}}{\sqrt{a_0}} \quad (10)$$

□

Corollary 1: If multiple links, that are part of some path, P, are removed the distortion of the path is bounded by $\sqrt{\rho}/\sqrt{\rho_m}$, where ρ_m is the smallest of the asymmetry constants of these links.

Proof: In lemma 1 we proved that $w_K(e)/w_{K^-}(e) \leq \sqrt{\rho}/\sqrt{\rho_e}$ for any link e. Thus:

$$\begin{aligned}\frac{\sum_{e\in P} w_K(e)}{\sum_{e\in P} w_{K^-}(e)} &\leq \frac{\sum_{e\in P} w_{K^-}(e)\sqrt{\rho}/\sqrt{\rho_e}}{\sum_{e\in P} w_{K^-}(e)} \\ &\leq \frac{\sum_{e\in P} w_{K^-}(e)\sqrt{\rho}/\sqrt{\rho_m}}{\sum_{e\in P} w_{K^-}(e)} \leq \frac{\sqrt{\rho}}{\sqrt{\rho_m}}\end{aligned} \quad (11)$$

□

Theorem 2: The maximum distortion of applying the square-root transformation and the link deletion on a directed clique is $\sqrt{\rho}$.

Proof: For $(u,\ v)$, the larger of two opposite links, we reduce the $(u,\ v)$-path cost by a factor of $\sqrt{\rho_{u,v}}$ when we multiply and take the square-root, and by lemma 1 we, at most, reduce the path cost by an additional factor of $\frac{\sqrt{\rho}}{\sqrt{\rho_{u,v}}}$ when we delete link $(u,\ v)$ and additional other links. As a result the distortion for the long link is at most

$$\frac{\sqrt{\rho}}{\sqrt{\rho_{u,v}}} \cdot \sqrt{\rho_{u,v}} = \sqrt{\rho}. \quad (12)$$

For $(v,\ u)$, the short link, the square-root transformation increases the path cost by a factor of $\sqrt{\rho_{u,v}}$. The link deletion process can only decrease the path cost to become closer to (but always higher than) $w(v,\ u)$. □

Bartal's tree increases the distance between two nodes by $\log b$ on the average. This means that the advertised cost between two border nodes can be at least a factor of $\sqrt{\rho}$ below the actual distance, due to the averaging of the two opposite links and the distortion of the link deletion, or at most a factor of $\sqrt{\rho_e}\log b$ above the actual cost due

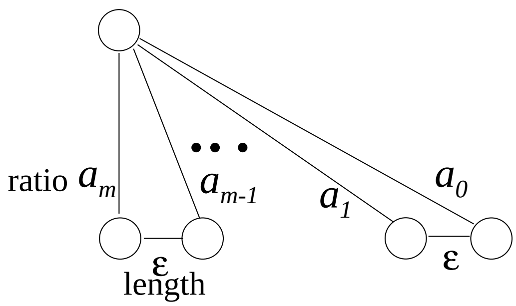

Fig. 3. Maximal cascading effect

to the averaging of the two opposite links and the tree construction.

The $(\log b)$-spanner algorithm [ADD+93] adds a link to the spanner only if the distance between the two link end points improves by $\log b$ as a result of the addition. Obviously, only links that belong to some shortest path are used, and specifically, links that are deleted to apply Bartal's tree construction do not effect the final result. The maximum $\log b$ distortion is measured on the shortest path between two nodes in $K^-(B)$. This was shown in theorem 2 to be at most $\sqrt{\rho}$. Thus, the distortion for the spanner is the same as the one for tree, i.e., $O(\sqrt{\rho} \cdot \log b)$.

IV. Simulation Results for the Link Deletion Procedure

To check the affect of the link deletion procedure, we produce random graphs according to Waxman's method [Wax88]. In the graph creation process, links are added until all nodes reach a minimum node degree of two. Links that increase node degree over 6 are rejected. This way the resulted graphs have both the clustering effect of close nodes and a relative high diameter. In a second phase, each undirected link is replaced by two directed link and the link weights are randomly assigned in the range $(1..10^{-6})$ uniformly on the log scale.

In tables I and II we present results for graphs with 100 nodes, of which 22 and 33 are border nodes, respectively. Figures 4 and 5 depict these graphs. Link weights were checked to cover the entire range and the average was around 11000, with variance around 3.0E+9. Link asymmetry ratios were ranging from just above 1 to over 500,000.

The percentage of deleted links varied from 38% to 70%, up to 31% of the triangles were bad, i.e., did not obey the triangle inequality. The additional distortion due to the link deletion process, i.e., due to the transformation of $K(B)$ to $K^-(B)$, is almost always below $\log b < 5$ (only 2 node pairs in the 8 experiments were distorted by more than 5), a negligible number when compared to $\sqrt{\rho} = 1000$.

In table III we present results for graphs with 100 nodes, 7 of which are border nodes. Between 2 and 10 of the 21 clique links where deleted, and up to 22 out of the 35 triangles where bad. Although the variance in several cases was very large, the additional distortion due to the link deletion process is below 4 as in table I.

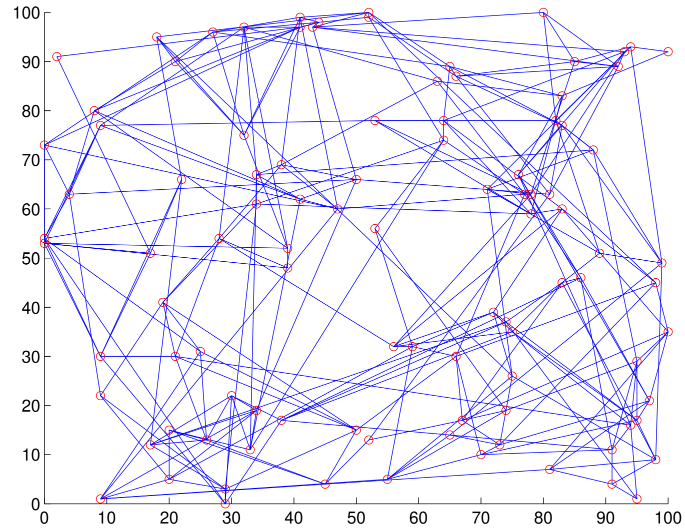

Fig. 4. A random graph with 100 nodes, 22 of which are border nodes.

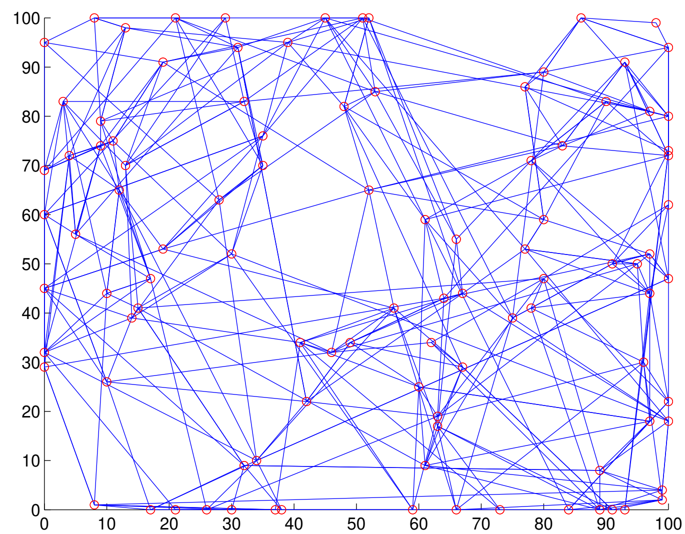

Fig. 5. A random graph with 100 nodes, 33 of which are border nodes.

V. Applicability to PNNI

In the PNNI standard [PNN96, section 3.3.8], a network ("complex node") is assumed to be comprised of undirected links and is aggregated by a star. A virtual node ("nucleus") is the interior reference point. For each state parameter (e.g., delay), a single default value is given to represent the value of this parameter for the connection between a border node and the nucleus ("spoke") in both directions.

To cope with asymmetry in the network structure, the standard allows "exceptions" of two types: a different value for a spoke of a certain border node, or a value for a "bypass" between two border nodes to represent a better cost of traversing the network between these two points. In guidelines that are supplied with the standard [PNN96, Appendix C] it is recommended that the number of excep-

border nodes asymmetry ratio				No. of	links	max.
max	min	ave	var	bad $\triangle$s	deleted	distortion
133.2	1.019	6.4	242	303	107	3.15
54.0	1.006	6.2	83	471	129	3.05
144.0	1.002	3.4	107	175	88	3.74
1284.1	1.002	21.5	8144	477	115	3.11

TABLE I
Results for a graph with 100 nodes, 22 of them are border nodes

border nodes asymmetry ratio				No. of	links	max.
max	min	ave	var	bad $\triangle$s	deleted	distortion
276.9	1.004	11.2	611	1546	341	4.95
413.1	1.002	5.2	418	823	305	4.88
864.3	1.001	8.9	1929	1101	372	5.70
100.2	1.011	4.0	45	767	338	2.72

TABLE II
Results for a graph with 100 nodes, 33 of them are border nodes

border nodes asymmetry ratio				No. of	links	max.
max	min	ave	var	bad $\triangle$s	deleted	distortion
2.98	1.1003	1.90	0.3	3	2	1.04
89.31	1.0370	9.49	336	10	9	1.99
1511.74	1.0386	173.85	131223	22	10	3.83
190.59	1.0214	24.75	2022	12	8	1.47

TABLE III
Results for a graph with 100 nodes, 7 of them are border nodes

tions used to configure a complex node will be kept smaller than $3b$, where b is the number of border nodes. There is no suggestion how to select exceptions.

This paper gives a framework for aggregating a directed "complex node" topology when the cost function is additive. Namely, given a directed network we suggest how to compress its representation, and in particular in a way that conforms with the PNNI standard. The aggregations described in the sequel give a bound on the worst case distortion in the advertised parameters which is $\log b \cdot \sqrt{\rho}$. The worst case distortion in the best star aggregation, using only the default spoke value, uses $\sqrt{D \cdot d}/2$ default spoke value and thus has a worst case distortion of $\sqrt{D/d}/2$, where D and d are the longest and shortest parameter values between pairs of border nodes, respectively. This means that for the case where the graph is undirected, our scheme distortion is only $O(\log b)$ while the star distortion remains $O(\sqrt{D/d})$ that might be very large. Note that, adding $3b$ exceptions in a naive way can not promise to fix the star distortion.

Our suggested aggregation is based on random Bartal trees. However, a tree that is built by applying Bartal's algorithm directly (see description below and the appendix) is comprised of not only the original network nodes (the border nodes in our case) but also of up to $b-1$ additional virtual nodes. The PNNI standard assumes one virtual node which is the star nucleus, and does not provide means to describe additional virtual nodes. To express the Bartal's tree aggregation in the PNNI nucleus-spoke plus exceptions framework, one must embed the tree in the links of $K^{-}(B)$ without loosing the logarithmic distortion. To describe our embedding, we must first provide a description of Bartal's algorithm.

Bartal's algorithm is comprised of two phases. In the first phase, the graph is recursively partitioned as follows. A node is arbitrarily selected, and all the nodes that are within a random radius from this node comprise the partition. The maximum value of the radius is a factor of k smaller than the diameter of the partition one level higher. This process continues for each partition, until all the nodes of the partition are in some sub-partition. For each sub-partition the process recurses.

In the second phase, a virtual node is assigned to each of the partitions in each level. Each virtual node, becomes the father of the virtual nodes that are assigned to the sub-partitions of its partition. The length of the links from a node to its children is half the partition diameter.

Our aim is to embed the virtual nodes in the network nodes in a way that does not increase the order of the distortion. After the partition phase, we pick a *center* for each partition P, i.e., we pick a node v whose maximal distance to any other node in the partition is minimal:

$$\max_{u\in P} w_{k^-}(v,\ u) = \min_{z\in P}\max_{u\in P} w_{k^-}(z,\ u)$$

The partition center radius is at most the partition diameter, which theoretically introduces a factor of two additional distortion. For the tree root we use the nucleus and the default spoke value for the embedding. In the worst case, when at every level a partition is divided to exactly two sub-partitions, $b-2$ nodes are embedded (the root is not embedded), and $2b-4$ bypass exceptions are used to represent the tree links. However, the selection of the node from which to start building a sub-partition is arbitrary. We choose to select the partition center to be the first node in the sub-partitioning. The resulted virtual tree contains almost no cycles and thus can be represented with close to $b-1$ bypass exceptions.

To further decrease the distortion, the weight of the link between two nodes in the tree is set to the weight of the shortest path between them, rather then the partition radius. Theoretically, this change does not decrease the average distortion, but, in practice, the distortion for some of the links decreases. Note that for the farthest nodes from the partition center, the factor two additional distortion may still apply due to the embedding of the virtual nodes in real nodes. However, it is not expected to be large in the high levels of the tree, where the distances are large, because at these levels, the number of nodes in a partition is fairly large and the center radius is expected to be close to half the partition diameter. An example of the construction of the Bartal tree and of our embedding is given in the appendix.

We suggest two structures based on the above trees. The first, a *t-tree*, is the union of t random Bartal's trees. For this purpose one might want to build more than t trees and pick the t combination that gives the best performance. A potentially better construction called quality partition forest (QPF) is a union of some highest level partitions. We note that all the random trees share the same nucleus, and the same high-level diameter, that is advertised as the default diameter. This gives rise to the following QPF construction. Build several trees, and select one of them in random to the aggregated graph. Add additional first-level sub-partitions according to some quality measure until the threshold of $3b$ exceptions is reached. The measure for the quality of a first-level sub-partition can be the maximal (or average) improvement of the distances over all pairs of node due to the addition of the sub-partition.

We tested the t-tree construction on the graphs of figures 4 and 5. Bartal's trees were iteratively created until the number of links in the union of the trees (exceptions) exceeded $2b$. Tables IV and V summarize the link distortions for the same random weight assignments that were used in tables I and II, respectively. The number of Bartal's trees calculated for the t-tree were 10, 12, 30, and 8 respectively, for table IV, and 7, 5, 25, and 7, respectively for table V. The average distortion is shown to be about 1.5 when the number of border nodes is 22, and about 2 when 33 border nodes are aggregated. However, more important are the histograms showing for a column i the number of node-pairs whose distance distortion is above $i-1$ but not above i. The histograms show that most of the node-pair distances are distorted by less than $\log b$, and a negligible fraction of the node pairs are distorted by more than $2\log b$.

The maximal distortions when a single tree is used for the random assignments used in table IV are 198, 13, 57.4, and 15.5. These distortions are 1-2 orders of magnitude higher than the maximum distortion of 6 obtained with the t-tree. In addition, the tail of highly distorted node-pair distances in the 1-trees is fat, meaning that many pairs are highly distorted. The maximal distortions for a 1-tree that correspond to the results in table V are 81.2, 26.6, 34.4, and 68.8, up to 10 times more than the results obtained by the t-tree. Thus, we can conclude that by joining several random trees, we successfully avoid the highly distorted node-pairs that are part of any single tree, and suggest a construction that is both implementable and with reasonable distortion.

Note, that the actual number of exceptions used in our t-tree is closer to $2b$ than to $3b$ (the last column in tables IV and V). Since the number of node-pairs whose distance cost is badly distorted is smaller than b, we can add a phase to the algorithm in which bypass exceptions between the badly distorted node-pairs are added, eliminating the thin tail in the histograms.

A. Simple Aggregation Algorithms

The t-tree construction, suggested above, is complex and hard to code. In this section, we test some simpler aggregation schemes that in theory may mal perform. However, in the experiments we conducted, one of these schemes, MST + 2 RST, exhibits exceptionally good performance. The good performance together with the ease of implementation make MST + 2 RST an attractive candidate as a heuristic method for practitioners. All the heuristics receive $K^-(B)$ as their input.

We tested the following methods:

- *MST* — the minimum spanning tree, comprised of $b-1$ edges.
- *3 RST* — The union of three random spanning trees. This structure can have up to $3b-3$ edges. In practice, since the random trees overlap we used about $2.5b$ edges.
- *MST + 2 RST* — The union of a MST and two random spanning trees. The maximal number of edges here is $3b-3$, the actual number of edges was similar to the 3RST case.
- *t-tree* — as described in the previous section. In most cases, the number of edges used was just over $2b$.

To compare between the aggregation schemes, we used the same graphs that were used above (figures 4 and 5). Figures 7 and 6 depict the cumulative histograms for the

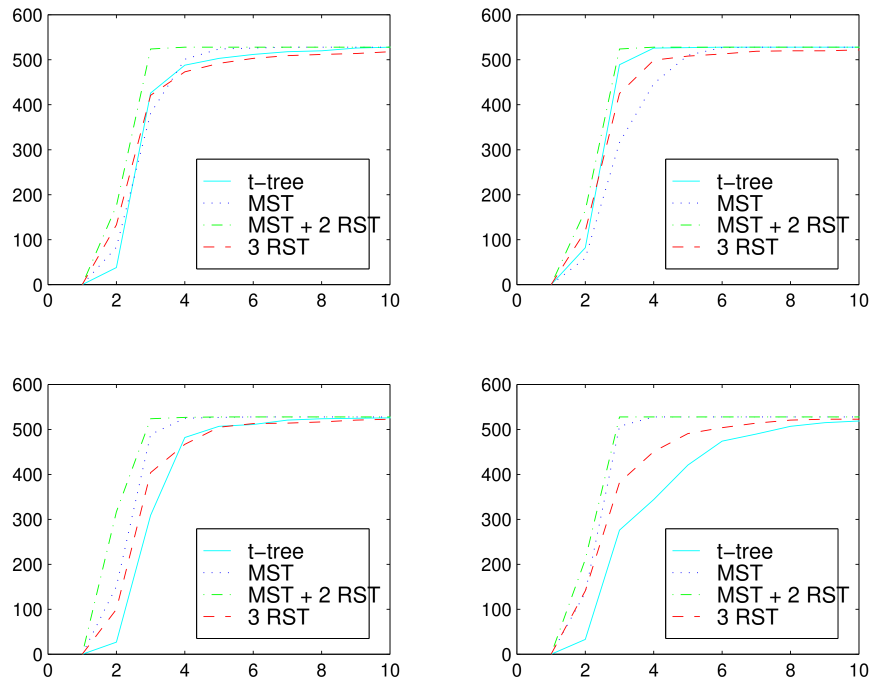

Fig. 6. A comparison between the histogram of the aggregation schemes for the 33 border node graph of figure 5.

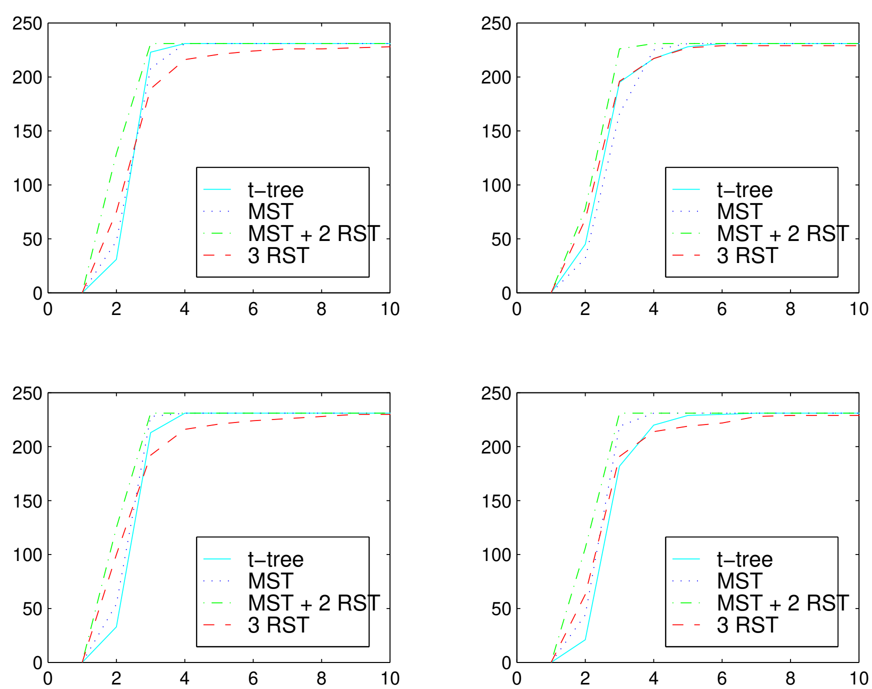

Fig. 7. A comparison between the histogram of the aggregation schemes for the 33 border node graph of figure 4.

			Histogram										links
max	ave	var	1	2	3	4	5	6	7	8	9	10	used
2.68	1.31	0.10	31	192	8	0	0	0	0	0	0	0	54
2.90	1.32	0.13	33	180	18	0	0	0	0	0	0	0	52
4.63	1.46	0.47	45	150	22	11	3	0	0	0	0	0	45
6.00	1.64	0.44	21	161	38	9	1	1	0	0	0	0	52

TABLE IV

Summary of the distortion for the graph of figure 4 (22 border nodes) when the t-tree structure is used for aggregation

			Histogram											links
max	ave	var	1	2	3	4	5	6	7	8	9	10	>10	used
8.83	1.78	1.21	38	388	62	15	9	6	2	6	2	0	0	72
11.18	2.01	1.25	27	282	173	25	4	10	3	1	1	0	2	67
4.39	1.38	0.17	82	407	37	1	1	0	0	0	0	0	0	67
23.72	2.75	4.64	33	243	68	77	53	16	17	8	4	3	5	67

TABLE V

Summary of the distortion for the graph of figure 5 (33 border nodes) when the t-tree structure is used for aggregation

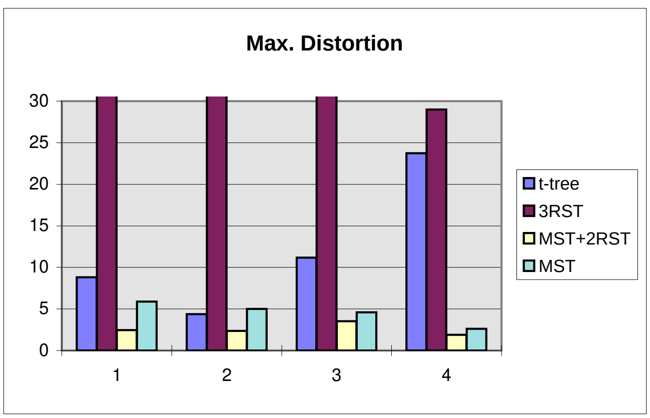

Fig. 8. A comparison between the maximum distortion of the aggregation schemes for the 33 border node graph of figure 5. For data sets 1, 2, and 3 the maximum distortion for 3RST is 249, 54 and 50, correspondingly.

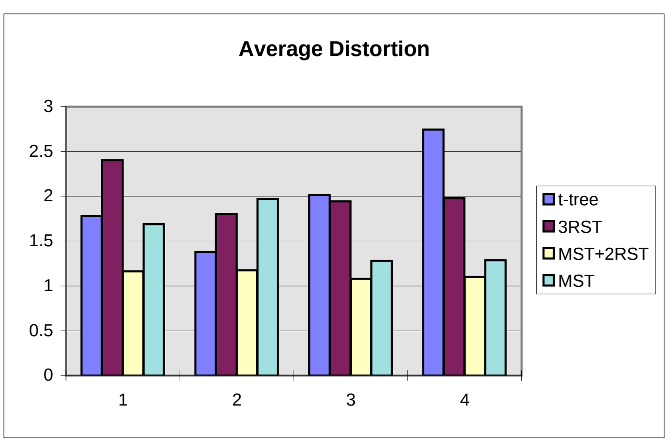

Fig. 9. A comparison between the average distortion of the aggregation schemes for the 33 border node graph of figure 5.

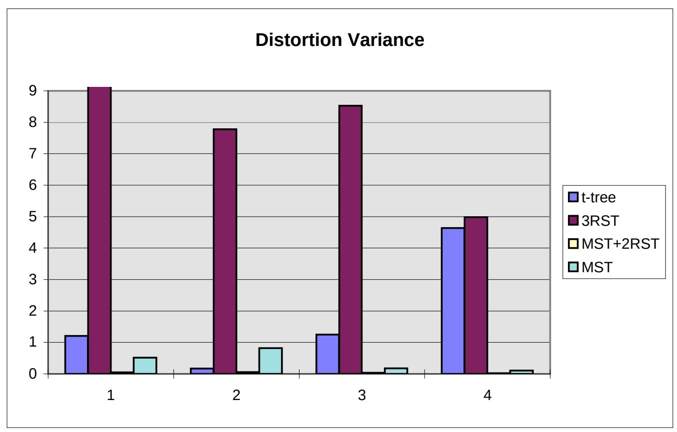

Fig. 10. A comparison between the variance of the distortion of the aggregation schemes for the 33 border node graph of figure 5. For data set 1 the variance of the distortion for 3RST is 122.

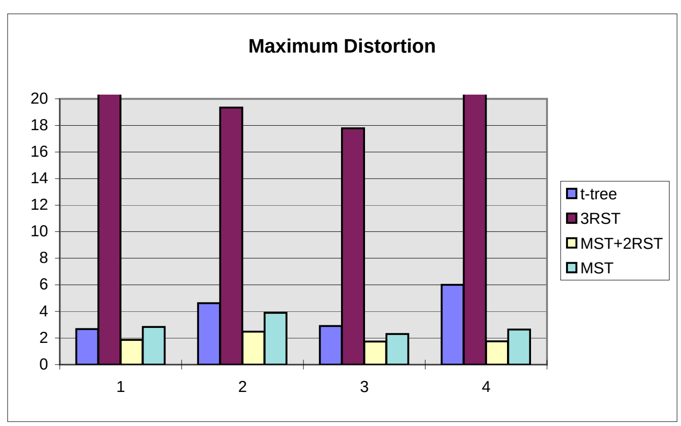

Fig. 11. A comparison between the maximum distortion of the aggregation schemes for the 22 border node graph of figure 4. For data sets 1 and 4 the maximum distortion for 3RST is 58.5 and 24.2, correspondingly.

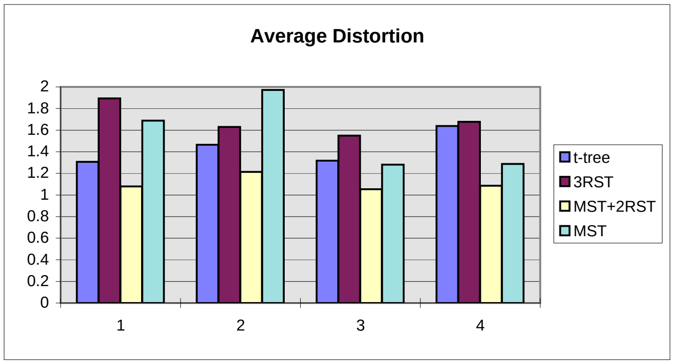

Fig. 12. A comparison between the average distortion of the aggregation schemes for the 22 border node graph of figure 4.

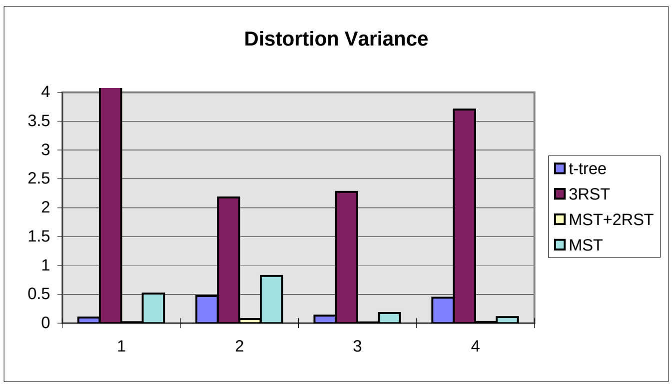

Fig. 13. A comparison between the variance of the distortion of the aggregation schemes for the 22 border node graph of figure 4. For data set 1 the variance of the distortion for 3RST is 16.2.

different aggregation schemes for four sets of experiments for each graph. The graphs show that MST has similar quality to the t-tree while using less than half the number of exceptions. MST with 2 RST has much tighter aggregation representation (with slightly more exceptions). In fact, in all but one experiment, this aggregation scheme had only a negligible number of node-pairs that were distorted by more than a factor of 2. 3 RST performs similar to the t-tree, in most cases, but since its aggregation distortion is higher (and sometime by a large margin) than MST + 2 RST and since the calculation of a MST is not harder than the calculation of a RST, there is no reason to prefer this scheme.

Figures 8 – 13 compare the maximum distortion, average distortion, and the variance of the distortion among the aggregation schemes. The average distortion of all the aggregation schemes is in the same range, but MST+2RST has the lowest average. The differences between MST+2RST and the rest of the aggregation schemes are between 20% and over 100%. However the maximum distortion graphs show that 3RST has a 1–2 orders of magnitude higher maximum distortion than the rest of the schemes. In some of the cases, t-tree has a maximum distortion that is several times higher than that of MST. This result in a higher variance in the distortion for 3RST and t-tree. The addition of two RST to an MST practically reduces the variance to negligible values.

VI. Concluding Remarks

We presented an aggregation scheme with a distortion that is theoretically bounded and show how it can be implemented in the ATM PNNI standard. We also presented a heuristic, based on a combination of minimum and random spanning trees that is shown to perform much better in practice, and is much simpler to implement.

Since once an aggregation is calculated, it is easy to check its quality, one can always use the low cost MST aggregation, augment it with RSTs if needed, and revert to t-tree in those cases where the aggregation distortion is not acceptable. Note that our results might be used also by practitioners when facing with the requirement to aggregate an undirected graph for standards such as PNNI.

It is interesting to compare our results with the ones obtained by Awerbuch et al. [ADKS98]. In their simulation study, they compared the overall performance of the PNNI routing algorithm when several aggregation schemes are employed. The aggregation schemes studied in [ADKS98] included MST and RST, which were compared to the case where no aggregation is used. MST was found to perform very well with little to no difference from the full knowledge case. This corresponds to our results that show that MST has low distortion.

Awerbuch et al. [ADKS98] found that the RST aggregation performance is sensitive to the topology, sometimes it is much worse than MST and sometime almost identical. This might be explained by the large variance in the distortion found in this study.

MST+2RST is shown here to be superior to MST. However, in [ADKS98] MST is shown to behave very well. It is interesting to investigate whether there are scenarios when the two aggregation schemes differ in practice in their effect on routing performance, or whether MST reaches some practical threshold performance beyond it more perfection of the aggregation has little effect on routing performance.

Acknowledgment

We would like to thank Amitabh Chaudhary and Amitabha Bagchi for their help in coding Bartal's algorithm.

References

[ADD+93] I. Althofer, G. Das, D. Dopkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete and Computational Geometry*, 9:81 – 100, 1993.

[ADKS98] Baruch Awerbuch, Yi Du, Bilal Khan, and Yuval Shavitt. Routing through networks with hierarchical topology aggregation. *Journal of High-Speed Networks*, 7(1):57 – 73, 1998.

[Bar96] Yair Bartal. Probabilistic approximation of metric space and its algorithmic applications. In *37th Annual IEEE Symposium on Foundations of Computer Science*, October 1996.

[CCS96] I. Castineyra, J. N. Chiappa, and M. Steenstrup. The nimrod routing architecture, February 1996. Internet Draft, Nimrod Working Group.

[GO99] Roch Guérin and Ariel Orda. QoS-based routing in networks with inaccurate information: Theory and algorithms. *IEEE/ACM Transactions on Networking*, 7(3):350 – 364, June 1999.

[Lee95] Whay Chiou Lee. Spanning tree method for link state aggregation in large communication networks. In *IEEE INFOCOM'95*, pages 297 – 302, April 1995.

[PNN96] Private network-network interface specification version 1.0 (PNNI). Technical report, The ATM Forum technical committee, March 1996. af-pnni-0055.000.

[PS89] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99 – 116, 1989.

[PU88] David Peleg and Eli Upfal. A tradeoff between space and efficiency for routing tables. In *20th ACM Symposium on the Theory of Computing*, pages 43 – 52, May 1988.

[Wax88] Bernard M. Waxman. Routing of multipoint connections. *Journal on Selected Areas in Communications*, 6:1617 – 1622, 1988.

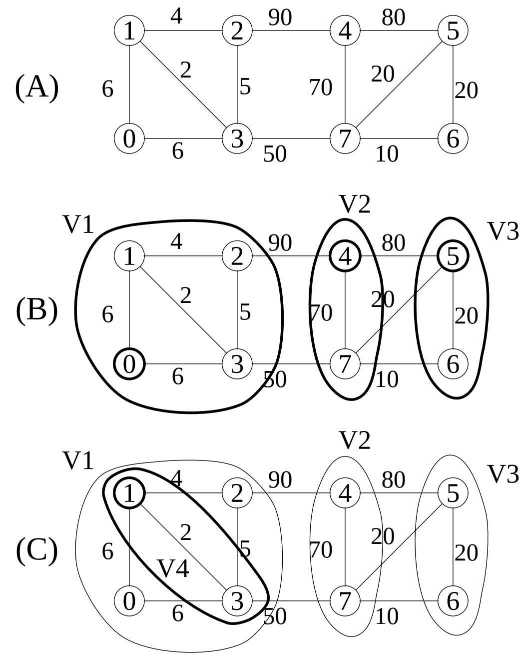

Fig. 14. The partition phase of Bartal's algorithm

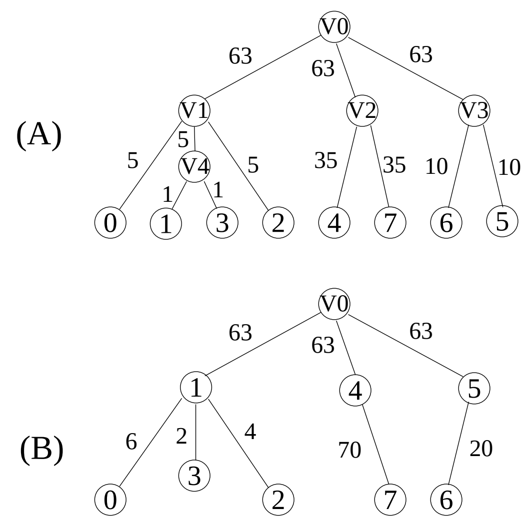

Fig. 15. Our embedding of Bartal's virtual tree in the network

Appendix

Appendix

An example of Bartal's algorithm and its implementation

Bartal's algorithm is comprised of two phases. In the first phase, the graph is recursively partitioned as depicted in figure 14. The network of figure 14(A) is partitioned by selecting an arbitrary node and a random radius and covering all the nodes within this radius from the selected node. Partition V1 is created by selecting node 0 with radius 53, partition V2 is created by selecting node 4 with radius 72, and partition V3 is created by selecting node 5 with radius 22.

Before further partitioning a partition P we calculate the partition center by selecting the node v whose maximal distance to any other node in the partition is minimal, i.e.,

$$\max_{u \in P} w_{k^-}(v,\ u) = \min_{z \in P} \max_{u \in P} w_{k^-}(z,\ u)$$

If the node used to build the partition is one of the nodes that match this criteria it is selected. In the example of figure 14(B) nodes 1, 4, and 5 are selected as the centers for partitions V1, V2, and V3, respectively.

Since to form a sub-partition we start with an arbitrary node, the partition center is always selected first. This reduces the number of virtual links used in the embedding step. Thus for partition V1 we start the subpartitioning with node 1, and as depicted in figure 14(C) it forms a subpartition with node 3. All the rest of the subpartitions are singletons.

In the second phase, a virtual node is assigned to each of the partitions in each level (see figure 15(A)). Each virtual node, becomes the father of the virtual nodes that are assigned to the sub-partitions of its partition. The length of the links from a node to its children is half the partition diameter. For example, the diameter of partition V1 is 10, and thus, the distance between node V1 and its children in the tree in figure 15(A) is 5.

Our aim is to embed the virtual nodes in the network nodes in a way that does not increase the order of the distortion. We replace each virtual node with the center of the partition it represents. If a parent and a child are replaced with the same node, e.g., node V1 and V4 in figure 15(A) are both replaced with node 1, they are merged to one and the link between them is deleted. The weights of the tree links is the shortest path between the two connected nodes (see figure 15(B)). Only the tree root (node V0 in our example) is not embedded in the network and is used as the nucleus. Its distance from the rest of the nodes is given by half the diameter, and is advertised as a default spoke.

Note that to advertise the resulted tree we need, in addition to the default spoke, only five bypass exceptions: links (0,1), (1,2), (1,3), (4,7), and (5,6).

Baruch Awerbuch Baruch Awerbuch is a professor (with tenure) at the Computer Science Dept. at Johns Hopkins University. He was a professor at MIT Mathematics Dept. from 1985 till 1992. His current Research interests include: Competitive Analysis of Online Algorithms, Distributed Computing, and Networks.

He has published more than 100 papers in journals and refereed conferences in the general area of design and analysis of networks and distributed systems. His prior work was primarily of theoretical nature, and involved designing algorithms for networks and distributed systems with provable guarantees on performance and reliability. His current work involves implementation of the some of the algorithms developed in the context of networks and distributed systems.

Baruch Awerbuch served a member of the Editorial Board for Journal of Algorithms and was a program chair of the 1995 ACM Conference on Wireless Computing & Communication and a member of the program committees of the ACM Conference on Principles of Distributed Computing (PODC) in 1989 and of the Annual ACM Symposium on Theory of Computing (STOC) in 1990 and 1991.

Yuval Shavitt Yuval Shavitt (IEEE S'88–M'97) received the B. Sc. in Computer Engineering (cum laude), M. Sc. in Electrical Engineering and D. Sc. from the Technion — Israel Institute of Technology, Haifa, Israel in 1986, 1992, and 1996, respectively.

From 1986 to 1991, he served in the Israel Defense Forces first as a system engineer and the last two years as a software engineering team leader. After graduation he spent a year as a Postdoctoral Fellow at the Department of Computer Science at Johns Hopkins University, Baltimore, MD. Since 1997 he is a Member of Technical Stuff at Bell Labs, Lucent Technologies, Holmdel, NJ. Recently Dr. Shavitt also joined the Department of Electrical Engineering - Systems, in Tel-Aviv University, Tel-Aviv, Israel. His current research focuses on active networks and their use in network management, QoS routing, and Internet mapping and characterization.