Dynamic Routing Schemes for Graphs with Low Local Density

AMOS KORMAN

Technion

AND

DAVID PELEG

Weizmann Institute

Abstract. This article studies approximate distributed routing schemes on dynamic communication networks. The work focuses on dynamic weighted general graphs where the vertices of the graph are fixed, but the weights of the edges may change. Our main contribution concerns bounding the cost of adapting to dynamic changes. The update efficiency of a routing scheme is measured by the time needed in order to update the routing scheme following a weight change. A naive dynamic routing scheme, which updates all vertices following a weight change, requires $\Omega(Diam)$ time in order to perform the updates after every weight change, where $Diam$ is the diameter of the underlying graph. In contrast, this article presents approximate dynamic routing schemes with average time complexity $\tilde{\Theta}(D)$ per topological change, where D is the local density parameter of the underlying graph. Following a weight change, our scheme never incurs more than $Diam$ time; thus, our scheme is particularly efficient on graphs which have low local density and large diameter. The article also establishes upper and lower bounds on the size of the databases required by the scheme at each site.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; E.1 [**Data**]: Data Structures—*Distributed data structures; graphs and networks*; G.2.1 [**Discrete Mathematics**]: Combinatorics; G.2.2 [**Discrete Mathematics**]: Graph Theory; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; F.2.3 [**Analysis of Algorithms and Problem Complexity**]: Tradeoffs between Complexity Measures

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Routing schemes, dynamic networks, distributed algorithms

---

This work was supported in part by grants from the Israel Science Foundation and the Israel Ministry of Science and Art.

Authors' addresses: A. Korman, Information Systems Group, Faculty of the IE&M, The Technion, Haifa, 32000 Israel. e-mail: pandit@tx.technion.ac.il; D. Peleg, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel, e-mail: david.peleg@weizmann.ac.il.

ACM Reference Format:

Korman, A. and Peleg, D. 2008. Dynamic routing schemes for graphs with low local density. ACM Trans. Algor. 4, 4, Article 41 (August 2008), 18 pages. DOI = 10.1145/1383369.1383372 http://doi.acm.org/10.1145/1383369.1383372

1. *Introduction*

1.1. MOTIVATION. The basic function of a communication network, namely, message delivery, is performed by its *routing scheme*. Subsequently, the performance of the network as a whole may be dominated by the quality of the routing scheme. Thus, constructing an efficient routing scheme is one of the most important tasks when dealing with communication network design.

We distinguish between *static* and *dynamic* routing schemes. In a static routing scheme the databases of the processors are tailored to the particular network topology. However, in most communication networks the typical setting is highly dynamic, that is, even when the physical infrastructure is relatively stable, the network-traffic load patterns undergo repeated changes. Therefore, for a routing scheme to be useful in practice, it should be capable of reflecting up-to-date load information in a dynamic setting, which may require occasional updates to the databases.

The communication network is modeled by an undirected connected graph. We consider the local distributed model in which the size of message is unbounded [Linial 1992; Peleg 2000]. The communication is asynchronous, and it is assumed that a message transmitted over a link arrives after an arbitrary, but finite, delay. For the sake of complexity analysis, we assume the slowest message to take at most one unit of time.

A simple dynamic routing scheme can be constructed which updates all vertices following each topological change. In order to perform the updates, such a scheme requires $\Omega(Diam)$ time after every weight change, where *Diam* is the diameter of the underlying graph. Ideally, upon a topological change, only a few time units should pass before the updates are completed. We distinguish between the *average time complexity* and *maximum time complexity* of a dynamic scheme π on a graph G, defined as follows. Given a scenario σ of topological changes on G, the average (respectively, maximum) time complexity of π during σ is the average (respectively, maximum) number of time units required by π to update the vertices following a topological change in σ. The average (respectively, maximum) time complexity of π on the graph G is the average (respectively, maximum) of this value over all scenarios of topological changes in G. As shown later, the maximum time complexity of a routing scheme is $\Omega(Diam)$ in many cases, even for graphs with constant local density. We therefore concentrate on constructing dynamic routing schemes which are efficient in terms of their average time complexity.

The efficiency of a dynamic scheme is measured not only by its time complexity, but also by the quality of the routes it provides and by the memory complexities associated with it. Route quality is measured by the *stretch factor* of the scheme, that is, the maximum ratio over all pairs of nodes in the network between the length of route provided for them by the routing scheme and the actual (weighted) distance between them in the network. We focus on β-*approximate* routing schemes, namely, those producing a route whose weighted length is perhaps not the shortest possible, but approximates it by a factor of at most β, for some constant $\beta > 1$.

Another consideration is the amount of information stored at each vertex. We distinguish between the *internal database* DATA(v), used by each node v to deduce the required information in response to online queries, and the additional external storage *Memory*(v) at each node v, used during (offline) updates and maintenance operations. For certain applications, the internal database DATA(v) is often kept in the router itself, whereas the additional storage *Memory*(v) is kept on some external storage device. Subsequently, the size of DATA(v) is a more critical consideration than the total amount of storage needed for the information maintenance.

The current article investigates routing schemes on dynamic settings involving changing link weights. The model studied considers a network whose underlying topology is a fixed graph, that is, the vertices and edges of the network are fixed but the (positive integer) weights of the edges may change. At each time, the weight of one of the edges can increase or decrease by a fixed quanta (which for notational convenience is set to be 1), so long as the weight remains a positive integer. (Our algorithms and bounds apply also for larger weight changes, since, clearly, a weight change of $\Delta > 1$ can be handled, albeit naively, by simulating it as Δ individual weight changes of 1. As our focus is on establishing the complexity bounds for the problem, no attempt was made to optimize the performance of our algorithms in case of large weight changes.)

We remark that it can be shown that if edge weights are also allowed to be zero, then on almost any underlying graph, the average complexity of any approximated dynamic routing scheme is $\Omega(Diam)$. Also note that if edges are allowed to be deleted and added, it is shown in Afek et al. [1989] that the average message complexity of constant-approximation routing schemes on general dynamic graphs is $\Omega(n)$. This lower bound also applies for the average time complexity, even on underlying graphs with constant local density, defined next.

For a graph G and integer $r \geq 1$, let $N(v, r)$ denote the set of vertices at distance at most r from the node v. Then the *local density* parameter of G is $D = \max_{v,r}\{|N(v, r)|/2r\}$. This graph-theoretic parameter is well studied, especially with relation to the bandwidth B of the graph G [Feige 2000; Feige et al. 2005; Chinn et al. 1982; Chung et al. 1989; Krauthgamer et al. 2004]. In fact, it is easy to show that $D \leq B$. Previous research suggests the conjecture that $B = O(D \log n)$, and it is known that $B = O(D \operatorname{polylog} n)$ [Feige 2000].

In this work we present dynamic approximate routing schemes that are particularly efficient on graphs having low local density and large diameter. Specifically, our dynamic approximate routing schemes have average time complexity $\tilde{\Theta}(D)$ on every underlying graph with local density D.

1.2. RELATED WORK. Many routing schemes and lower bounds for the resources required for routing were presented in the past [Peleg 2000]. The first studies attempting to characterize and bound the resource tradeoffs involved in routing schemes for general networks were presented in Peleg et al. [1989], Awerbuch et al. [1992, 1990] and were followed by a number of improved constructions [Cowen 2001; Iwama et al. 2003, 2000; Thorup et al. 2001]. These studies focused on routing schemes with compact routing tables and low stretch factors.

Unfortunately, most of the known algorithms in this field apply only for static networks, and only a few papers consider dynamic networks. In Santoro et al. [1985] a partial solution is presented for some limited cases of topology changes that keep the network in a tree topology.

The following dynamic routing schemes on trees assume the designer of the routing scheme to have freedom to choose the identities of the vertices. Afek et al. [1989] presents a routing scheme for the restricted case of dynamic growing trees using identities of size $O(\log^2 n)$, as well as database size $O(\Delta \log^3 n)$, where Δ is the maximum degree in the graph, and average message complexity $O(\log n)$ per topological change, where n is an upper bound on the number of vertices in the growing tree. When an upper bound n on the number of vertices in the growing tree is known in advance, a routing scheme with average message complexity $O(\frac{\log n}{\log \log n})$ and polylogarithmic database size is given in Korman et al. [2004]. In the more general case where in each step, a leaf can either be added to or removed from the tree, a routing scheme with $O(\log^2 n)$ average message complexity and $O(\log^2 n)$ database size is presented in Korman et al. [2004] and a routing scheme with $\Theta(\log n)$ database size and sublinear average message complexity is presented in Korman [2005]. All the aforementioned dynamic routing schemes deal only with tree networks. For general graphs there are even fewer results. In the case where edges may be added to or deleted from the graph, a lower bound of $\Omega(n)$ is established in Afek et al. [1989] for the average message complexity of constant-approximation routing schemes on general dynamic graphs. This can be translated to a lower bound of $\Omega(n)$ on the average time of constant-approximation routing schemes on general dynamic graphs. Moreover, this bound holds even on graphs with constant local density. In addition, another lower bound of $\Omega(n)$ is established in Afek et al. [1989] for the average message complexity of constant-approximation routing schemes on growing trees, in the case where the adversary of the routing scheme chooses the identities of the vertices.

The routing scheme of Dolev et al. [1999] for dynamic graphs applies to a somewhat different networking model based on virtual circuits, and route quality is measured therein in terms of the number of “super-hops” required for a route; hence, those results cannot be directly compared with ours. Also, the analysis therein does not consider the length of routes produced by the routing scheme, and in fact the scheme may incur a linear stretch factor in some dynamic scenarios.

The maximum database size of β-approximate point-to-point routing schemes (defined next) on general graphs was previously investigated for the static scenario. A lower bound of $\Omega(n^{\frac{1}{2\beta+4}})$ for $\beta \geq 2$ was shown in Peleg et al. [1989] and of $\Omega(n \log n)$ for $1 \leq \beta < 2$ in Fraigniaud et al. [1997].

In the *sequential* (nondistributed) model, dynamic data structures have been studied extensively. For comprehensive surveys on dynamic graph algorithms, see Eppstein et al. [1999] and Feigenbaum et al. [2000].

1.3. The Model.

In this article the underlying network topologies considered are general graphs. Throughout the work, we denote by n the number of vertices in the network. This research studies two types of routing scheme. *Source-directed* routing schemes are routing schemes in which the message originator computes the entire route to the destination and attaches it to the message header. In contrast, *point-to-point* routing schemes route messages on a hop-by-hop basis, with each intermediate node along the the route determining the next edge to be used.

Formally, we make the following definitions. A *point-to-point β-approximate routing scheme* is composed of an *update protocol* for assigning each vertex v of the graph with a local database $\text{DATA}(v)$, coupled with a *router* algorithm whose inputs are the header of a message M, DATA(v) and the identity of a vertex u. If a vertex x wishes to send a message M to vertex y, it first prepares a header for M and attaches it to M. Then, x's router algorithm outputs a port of x on which the message is delivered to the next vertex, until it reaches y. The requirement is that the weighted length of the resulting path connecting x and y is a β-approximation of the weighted distance between x and y at the time the route starts. A *source-directed β-approximate routing scheme* is composed of an *update protocol algorithm* for assigning each vertex v of a graph with a local database DATA(v), coupled with a router algorithm which, using only that information in DATA(v) and the identity of a vertex u, outputs a sequence of port numbers. This sequence represents a path connecting v and u whose weighted length is a β-approximation of the distance between u and v.

1.4. OUR CONTRIBUTION. Our main contribution focuses on the average time complexity of dynamic routing schemes. We use the local density parameter in an attempt at capturing the graph-theoretic parameter governing the message complexity of the problem.

Let $\beta > 1$ be constant. In Section 3 we present our β-approximate source-directed scheme for the class $\mathcal{F}(n, D)$ of n-vertex graphs with local density at most D. In Section 4 we present our β-approximate point-to-point routing scheme for $\mathcal{F}(n, D)$. Both schemes incur an average time complexity $O(D \log^2 n)$ per weight change.

In terms of database size, we show an upper bound of $O(n \log n)$ and a lower bound of $\Omega(n)$ for database size of a β-approximate source-directed routing scheme on $\mathcal{F}(n, D)$, for $D \geq 2$. Our point-to-point β-approximate routing scheme uses database size $O(n \log^2 n)$. The point-to-point version uses a header size of $O(\log n)$ bits.

2. *Preliminaries*

An *unweighted* graph is one whose edges have unit weight. Denote by $d_G(u, v)$ the unweighted distance between u and v in G. For $q > 0$, the q-neighborhood of $v \in G$, denoted $\Gamma_G(v, q)$, is the subgraph of G induced by $\{u \mid d_G(v, u) \leq q\}$. The subscript G can be omitted when the graph G is clear from the context. It is assumed that the nodes of a given n-vertex graph have distinct identities in the range $1, \ldots, n$. The identity of a node v is denoted by $id(v)$.

The communication network is modeled by an undirected connected graph. We consider the local distributed model, in which the size of the messages is unbounded [Linial 1992; Peleg 2000]. The communication is asynchronous, and it is assumed that a message transmitted over a link arrives at an arbitrary, but finite, delay. For the sake of complexity analysis, we assume that the slowest message takes at most one unit of time.

2.1. THE DYNAMIC NETWORK MODEL. We assume that the vertices and edges of the network are fixed and that the edges of the network are assigned positive integer weights. In the dynamic network model considered in this article, it is assumed that the weights of the network links may change from time to time. For example, the weights may represent the current loads on the links, which may change due to queue buildups and nonuniform arrival rates. For two vertices u and v in a weighted graph G with edge-weight function ω and for a time t, denote by

$d^{\omega}_G(u, v, t)$ the weighted distance between u and v at time t. The following dynamic events may occur from time to time.

(1) *Positive Weight Change*. An edge $e = (u, v)$ increases its weight by one.
(2) *Negative Weight Change*. An edge $e = (u, v)$ with weight $w(e) \geq 2$ decreases its weight by one.

Subsequent to an event on an edge $e = (u, v)$, its endpoints u and v are informed of this event.

2.2. Routing Schemes.

2.2.1. *Static Routing Schemes.*

Let us first describe our routing schemes in the static setting. For fixed $\beta > 1$, a static *β-approximate source-directed routing scheme $\pi_{SD} = \langle \mathcal{P}_{SD}, \mathcal{R}_{SD} \rangle$* for a family of graphs $\mathcal{F}$ is composed of the following components.

(1) A *preprocessing* algorithm $\mathcal{P}_{SD}$, given a graph $G \in \mathcal{F}$, assigns a local database DATA(v) to each vertex $v \in G$.
(2) A polynomial-time router algorithm $\mathcal{R}_{SD}$, given the database DATA(u) and $id(v)$ for some vertices u and v in some graph $G \in \mathcal{F}$, outputs a sequence of port numbers representing a path P connecting u and v.

A *static β-approximate point-to-point routing scheme $\pi_{PTP} = \langle \mathcal{P}_{PTP}, \mathcal{R}_{PTP} \rangle$* for a family of graphs $\mathcal{F}$ is composed of the following components.

(1) A preprocessing algorithm $\mathcal{P}_{PTP}$, given a graph $G \in \mathcal{F}$, assigns a local database DATA(v) to each vertex $v \in G$.
(2) A polynomial-time router algorithm $\mathcal{R}_{PTP}$, given the database DATA(u), and $id(v)$ for some vertices u and v in some graph $G \in \mathcal{F}$, and a header H of a message M, outputs a port number of u and a new header for M.

Routing a message using a point-to-point routing scheme π_{PTP} is done as follows. If a vertex x wants to send a message M to the vertex y in G, it first prepares a header H for M and attaches it to M. Then the message is delivered via the port $\mathcal{R}_{PTP}(Data(x), id(y), h)$ to x', a neighbor of x. The vertex x' repeats the process, using its own database DATA(x'). The message M is thus delivered on the port $\mathcal{R}_{PTP}(Data(x'), id(y), h)$ of x' to the next vertex, and so forth. In contrast, when using a source-directed routing scheme π_{SD}, if a vertex x wants to send a message M to the vertex y in G, then x computes a path P connecting x and y and attaches it to the message header. Each vertex on the route delivers M to the next vertex on the path P until M reaches its destination y. For the routing scheme (either source-directed or point-to-point) to be β-approximate, the requirement is that the weighted length of the resulting path connecting x and y is a β-approximation of $d^{\omega}_G(x, y)$.

Note that if the number of bits in the header is not bounded, then any source-directed scheme can be transformed into a point-to-point routing scheme, simply by writing the output path into the header. Typically, it is assumed that on n-node graphs, the number of bits in the header is bounded by $O(\log n)$.

2.2.2. *Dynamic Routing Schemes.* In the asynchronous dynamic network model, the preprocessing algorithm $\mathcal{P}$ changes into an update protocol $\mathcal{U}$ (denoted $\mathcal{U}_{SD}$ for source-directed routing and $\mathcal{U}_{PTP}$ for point-to-point routing) that initially assigns a local database $\textsc{Data}(v)$ to each vertex $v \in G$. After the initial setup, $\mathcal{U}$ is activated after every change in the network topology, in order to maintain the databases of all vertices in the underlying graph so that the corresponding router algorithms work correctly. Observe that in the context of distributed networks, the update algorithms must be implemented as *distributed update protocols*. In particular, the messages sent by $\mathcal{U}$ in order to maintain the databases are sent over the edges of the underlying graph.

It is easier to analyze our protocols assuming that the topological changes occur sequentially and are sufficiently spaced, so that the update protocol has enough time to complete its operation in response to a given topological change before the occurrence of the next change. However, our schemes can operate also under weaker assumptions. Specifically, it is allowed for topological changes to occur in rapid succession, or even concurrently. Our statements concerning the correctness of our source-directed routing scheme (a scheme is correct if every message sent will eventually reach its destination) still hold. Our point-to-point scheme is also correct, provided that the topological changes quiet down at some later time for a sufficiently long time period. The quality of our schemes, however, is affected by such behavior of the system in the following way. We say that a time t is *quiet* if all updates (of the relevant update protocol) concerning the previous topological changes have occurred by time t. At a quiet time t, the system is said to be quiet. Our demand from a dynamic β-approximate routing scheme (either source-directed or point-to-point) is that if a route from u to v starts at some quiet time t and the system remains quiet throughout the rest of the route, then the weighted length of the resulting route is a β-approximation to $d^{\omega}(u, y, t)$. The aforesaid demand is reasonable, as it can be easily shown that for any routing scheme, if a route from u to v starts at a time t, where t is not a quiet time, then we cannot expect the resulting route to be a β-approximation to $d^{\omega}(u, v, t)$ for any $\beta > 1$.

Let $\pi(\beta) = \langle \mathcal{U}(\beta), \mathcal{R} \rangle$ be a dynamic β-approximate routing scheme on the family $\mathcal{F}$. Given a scenario σ of topological changes on some graph $G \in \mathcal{F}$, the average (respectively, maximum) time complexity of $\pi(\beta)$ during σ, denoted $\textsc{Avg\_Time}(\pi(\beta), \sigma)$ (respectively, $\textsc{Max\_Time}(\pi(\beta), \sigma)$), is the average (respectively, maximum) number of time units required by $\pi(\beta)$ to update the vertices following a topological change in σ. We are interested in the following complexity measures.

—*Maximum Database Size*, $\textsc{Data}(\pi(\beta))$. This is the maximum number of bits in $\textsc{Data}(v)$ taken over all vertices v and all scenarios on all n-vertex graphs $G \in \mathcal{F}$.

—*Average Time Complexity*, $\textsc{Avg\_Time}(\pi(\beta))$. This consists of the maximum value of $\textsc{Avg\_Time}(\pi(\beta), \sigma)$ over all scenarios σ on all n-node graphs $G \in \mathcal{F}$.

—*Maximum Time Complexity*, $\textsc{Max\_Time}(\pi(\beta))$. This is the maximum value of $\textsc{Max\_Time}(\pi(\beta), \sigma)$ over all scenarios σ on all n-node graphs $G \in \mathcal{F}$.

For the point-to-point routing scheme, we are also interested in the following complexity measure.

—*Header Size*, $\mathcal{HD}(\pi_{PTP})$. This measure is the maximum number of bits attached to the message header by the router protocol π_{PTP} at any step along the route.

2.3. GENERAL INTUITION. Our schemes are based on the following ideas. After every change in a weight of an edge e, one of e's end-nodes creates a "report" of this event, encoded on a token that is sent to some vertices in the graph. A simple routing scheme would require the update protocol to send each such token to all vertices. For such a scheme, on the one hand, all nodes have an up-to-date view of the graph and the routings can be made over the shortest weighted paths, but on the other hand, the time complexity of the update protocol is high, namely, $\Omega(Diam)$ time units required after *every* topological event.

In order to reduce the time complexity, both our schemes are based on the principle that updates are made in a gradual manner: Tokens are disseminated only to a limited distance and are then stored in intermediate bins of various sizes. Nodes outside this range are thus unaware of the changes represented by these tokens. Whenever sufficiently many tokens accumulate at a bin, they are disseminated further, to a distance proportionate to their number. The analysis of the approximation is based on bounding the possible overall error made in that path selected by the router protocol. This bound is based on the maximal distortion in the way that relevant vertices view the weights of edges in some subgraph, in comparison to the reality. This distortion corresponds to the number of delayed tokens "stuck" in the various bins of this subgraph.

Algorithms based on the idea of gradual token-passing appeared in Awerbuch et al. [1996] and Korman et al. [2003, 2004]. However, a direct application of the method presented in the aforementioned papers would not yield a routing scheme with the desired properties. Informally, the reason is that the algorithms used in the these papers were designed for trees. Moreover, using techniques similar to those in the previous papers, we can only guarantee that each node x knows an approximation of the weighted length of any path *in some fixed spanning tree* containing x, while for our purposes we are interested in approximations of all paths *in the graph* that pass through x. Hence, we extend the techniques of the preceding papers by separating the updates, which are made on the graph, from the token passing, which is carried out on the spanning tree. The token passing implicitly monitors the updates. Each time a bin b becomes full, it is emptied and its contents are used to update vertices on the graph at distance $d(b)$ proportionate to the size of b. In addition, the contents of b are transferred to b', a bin on the spanning tree, at distance $d'(b)$ (which is also proportionate to the size of b). We note that simply letting $d(b) = d'(b)$ would not work, as the path approximations turn out to be insufficient. Our performance bounds therefore rely on letting $d(b) = c \cdot d'(b)$ for some constant $c > 1$.

When routing a message from x to y in our source-directed routing scheme, x outputs $\mathcal{R}_{SD}(Data(x), y)$, the shortest path from x to y (according to x's knowledge). It will follow from our analysis that $\mathcal{R}_{SD}(Data(x), y)$ is a good approximation to $d^{\omega}(x, y)$.

However, things turn out to be more difficult in the point-to-point routing scheme. A natural approach for constructing the point-to-point scheme would be to use the same data structure as in the source-directed scheme, except that whenever v receives a message addressed to y, v delivers the message to the next node (i.e., its neighbor) on the path $\mathcal{R}_{SD}(Data(v), y)$. Unfortunately, this may cause the routing process to end up with a message caught in an infinite loop. For example, since

DATA(v) and DATA(w) are not identical, v may think that w is the next node on the shortest path from v to y, and w may think that v is the next node on the shortest path from w to y.

The main technical contribution of this article is based on the following idea, which is used by our point-to-point routing scheme in order to prevent the aforementioned undesirable phenomenon. When routing a message from x to y, x first estimates $d^{\omega}(x, y)$ as in the source-directed scheme, and uses this estimate to define some value $q = \Theta(d^{\omega}(x, y))$ that will be attached to the message header. When getting a message destined for y, the intermediate node v along the route creates a collection $\tilde{\Gamma}_v(y, q)$ of estimates for the weights of all edges in the q-neighborhood of y, $\Gamma(y, q)$. As established later in our present work, the estimations $\tilde{\Gamma}_v(y, q)$ have two important properties. The first is that these estimates are the same for all vertices v on the route. This property allows each intermediate node v along the route to mimic the shortest path computation carried by x in order to decide to which node it should pass the message, yielding consistency between the decisions of nodes on the route. The second property is that although these estimates are potentially weaker than the corresponding estimates obtained by the source-directed routing scheme, they are still good enough to ensure that the route is a β-approximation to $d^{\omega}(x, y)$.

3. *The Source-Directed Routing Scheme $\pi_{SD}(\beta)$*

Let $\beta > 1$ be constant. In this section we introduce our β-approximate source-directed routing scheme $\pi_{SD}(\beta)$ for the family $\mathcal{F}(n, D)$. Ideas and tools developed in this section are used again in the next section for our more significant (and somewhat more sophisticated) point-to-point routing scheme $\pi_{PTP}(\beta)$.

3.1. GENERAL STRUCTURE. Let $T(G)$ be a spanning tree of some graph $G \in \mathcal{F}(n, D)$, rooted at some vertex r. Let $T'(G)$ be the tree obtained from $T(G)$ by extending it with an imaginary n-node path P_r attached to r. Let r' be the end node of P_r not attached to $T(G)$. We view r' as the root of $T'(G)$.

A token-passing mechanism is used in order to monitor the message delivery mechanism of the update protocol. In the token-passing mechanism, messages are sent up the imaginary tree $T'(G)$. As described later, in practice, the token-passing mechanism on $T'(G)$ is simulated on $T(G)$, by letting r, the root of $T(G)$, simulate the behavior of vertices in the path P_r. Informally, the reason for invoking the token-passing mechanism on $T'(G)$, rather than on $T(G)$, is to allow the delivery of tokens to continue beyond r, without necessarily being truncated at r. This enables ease of presentation of the update protocol by avoiding cumbersome descriptions of the behavior of r as a distinguished node.

The token-passing mechanism is a slight modification of the algorithms in Awerbuch et al. [1996] and Korman et al. [2004]. Still, some different parameters are used by our algorithm, and the delicacy of this token-passing mechanism helps to understand the delivery mechanism of the update protocol. Therefore, we now give a full description of our update protocol.

Each node v maintains two bins, a "local" bin b_l and a "global" bin b_g, storing a varying number of tokens throughout the execution. In the token-passing mechanism, whenever a weight change occurs at an edge (v, u), one of these nodes, say, u, adds a token to its local bin so as to indicate this event. When a bin gets filled with tokens, the latter are disseminated up the tree only to a limited distance and are then stored in an intermediate global bin of larger size. In our update protocol, whenever a bin b gets filled with tokens, in addition to being sent up the tree, the tokens are also sent to vertices of G, at a limited distance from b. Upon receiving these tokens, the vertices update their view of the graph accordingly; however, these vertices do not store these tokens.

Each token contains information about some weight change in one of the edges. Specifically, a token is of the form $\phi = \langle id(e), \omega(e), c\rangle$, indicating that the cth weight change on the edge e sets its weight to $\omega(e)$. In the following discussion, unless it might cause confusion, we do not distinguish between a bin and the node holding this bin. Let $H(v)$ denote v's unweighted (hop) distance from r'. For every node v of $T'(G)$, apart from r', the bins b_l and b_g are assigned a *level*, defined as follows.

$Level(b_g) = \max\{i \mid 2^i \text{ divides } H(v)\}$; and
$Level(b_l) = -1$.

Note that the level of the bin determines whether it is of type b_l or b_g. Therefore, in the following discussion we omit the subscripts g and l unless it might cause confusion. For each bin b at node v, the closest ancestor bin in $T'(G)$ (including possibly in v itself), b', satisfying $Level(b') = Level(b)+1$, is set to be the *supervisor* of b. If there is no such bin, then the global bin of r' is set to be the supervisor of b. Let $sup(b)$ denote the supervisor bin of b. Note that the supervisor bin of a local bin is either the global bin of the same node or the global bin of its parent in $T'(G)$. The supervisor relation defines a bin hierarchy. The following observation is given in Awerbuch et al. [1996] and in Korman et al. [2004].

OBSERVATION 3.1. (1) *The highest level of the bin hierarchy is at most* $\log n + 2$.
(2) *If* $Level(b) = l$*, then the path from* b *to* $sup(b)$ *has at most* $3 \cdot 2^l$ *nodes.*

For $\beta > 1$, let

$$\alpha = \frac{1+\sqrt{2\beta-1}}{2}, \quad \delta = \min\left\{\sqrt{\beta}-1, \frac{\alpha-1}{\beta\alpha}\right\}.$$

The number of tokens stored at each bin b at a given time is denoted $\tau(b)$. The *capacity* of each bin depends on its level. Specifically, a bin b on $Level(b) = l$ may store $0 \le \tau(b) \le Cap(l)$ tokens, where $Cap(l) = \max\{2^{\lfloor \log \frac{\delta \cdot 2^l}{6D\cdot(\log n+2)}\rfloor}, 1\}$.

3.2. THE UPDATE PROTOCOL, $\mathcal{U}_{SD}(\beta)$. The memory structure *Memory*(v) of each vertex v contains the adjacency matrix $A(v)$ of the entire initial graph. For each edge e, the counter $c(e)$ counts the number of changes in e's weight and is initially set to be 0. Each entry e in $A(v)$ contains two fields, denoted by $\omega(e, v)$ and $c(e, v)$. If the latest change in e's weight that v heard about was the c_0th change, which has led to the values $c(e) = c_0$ and $\omega(e) = \omega_0$, then e's entry in $A(v)$ is set to $\langle\omega(e, v), c(e, v)\rangle = \langle\omega_0, c_0\rangle$. In addition, the memory structure of v contains the tokens that v received from other nodes.

For each edge e, one of its endpoints (say, the one with the smaller id) is said to be *responsible* for e. A token $\phi = \langle id(e), \omega(e), c\rangle$ is said to be *fresh* with respect to the matrix $A(v)$ if c is larger than $c(e, v)$. Intuitively, such a token can be used to update the entry corresponding to e in $A(v)$.

Let $b_g(v)$ denote the global bin of a vertex $v \in T'(G)$. Let $T^*(G)$ be the same as $T(G)$, with the same bin hierarchy, except that r has a number of additional global bins. Specifically, the set of added global bins of r is $\{b_g(w) \mid w \in P_r\}$. We next describe the imaginary protocol Bin on $T'(G)$. In practice we run protocol Bin* on $T^*(G)$. This is the same as protocol Bin, except that r uses its multiple global bins to simulate the behavior of the imaginary path P_r in protocol Bin. For a level l bin b, let $Q(b)$ denote the set of all nodes at unweighted distance at most $2^5 \cdot 2^l$ from b (in G). Note that by the second item in Observation 3.1, $sup(b) \in Q(b)$.

The update protocol $\mathcal{U}_{SD}$ uses protocol Bin, described next, to maintain the databases of the vertices.

Protocol Bin

(1) Initially all bins are empty.
(2) For an edge e under u's responsibility, each time u learns that the weight $\omega(e)$ has increased or decreased by one, it adds $+1$ to $c(e, u)$ and adds a token to u's local bin, making it full. This token is a triplet $\langle id(e), \omega(e), c(e)\rangle$ where the first field contains the identity of the edge e, the second field contains $\omega(e)$, the new weight of the edge e, and the third field contains the value of the counter $c(e) = c(e, u)$.
(3) Whenever a bin b on level l gets filled with tokens, the following happens.
    (a) The bin b is emptied and its content is broadcast to all nodes in $Q(b)$. The broadcast is performed using a bounded-depth flooding protocol, where each message contains the content of b along with a counter $\rho(b)$ which is initially set to zero. Whenever the message reaches a node u over some link, the node increments the counter $\rho(b)$ by one and forwards the message over all other links, unless the counter has reached $2^5 \cdot 2^l$, in which case it does not forward it.
    (b) If the $sup(b) \neq r'$ then $sup(b)$ adds the content of b to itself. (Note that if $sup(b) = r'$ then it does not keep these tokens.)
    (c) Each node $z \in Q(b)$ updates its database as follows. Let $\phi = \langle id(e), \omega(e), c\rangle$ be a token in b. If ϕ is fresh with respect to the adjacency matrix $A(z)$ then z updates e's entry in $A(z)$ to be $(\omega(e), c)$.

3.3. The Data Structure Data(v) and the Router Protocol $\mathcal{R}_{SD}(\beta)$.

Let $A'(v)$ be the graph obtained by $A(v)$, where the weight of an edge in $A'(v)$ is $\omega(e, v)$, the first field in the e's entry of $A(v)$. Using an algorithm similar to the Dijkstra or the Bellman-Ford algorithms on the graph $A'(u)$, let Data(v) be the BFS tree of $A'(v)$ rooted at v. We therefore obtain the following lemma.

Lemma 3.2. Data$(\pi_{SD}(\beta)) = O(n \log n)$.

Given Data(u) and $id(v)$, $\mathcal{R}_{SD}(\beta)$ outputs the simple path P connecting u and v in the tree Data(u).

3.4. Analysis

3.4.1. *Correctness and Approximation.* Since the output of $\mathcal{R}_{SD}(\beta)$ $(Data(u), id(v))$ is a simple path connecting u and v, a route along this path starting at u will eventually reach v. It remains to bound the approximation ratio of the scheme.

We say that the token $\phi = \langle id(e), \omega(e), c\rangle$ corresponds to the cth weight-change event of e, which created the token ϕ in step 2 of protocol Bin.

Claim 3.3. *A bin b that holds a token ϕ, corresponding to an event occurring at distance at least 2^l from b on $T(G)$, must be of level at least $l - 3$.*

PROOF. Let v be such that the event corresponding to the token ϕ occurred in an edge under v's responsibility. The token was then created and stored at b^{-1}, v's local bin of level -1. Let $b^{-1}, b^0, b^1, b^2, \ldots, b^k$ be a sequence of bins such that $b^k = b$, for $0 \le i \le k$, b^i is the supervisor bin of b^{i-1} and at some time $t^{i-1} < t^i$, b^{i-1} contained ϕ and became full. By the definition of supervisors, b^i must be of level i, therefore we want to show that $k \ge l-3$. By the second item in Observation 3.1, the distance between b_i and b_{i+1} in $T(G)$ is at most $3 \cdot 2^i$, and by the assumption of the claim the distance between b^{-1} and b^k is at least 2^l. Therefore $3\sum_{i=-1}^{k} 2^i \ge 2^l$, yielding $k \ge l-3$. □

Fix a quiet time t. Let ϵ be an event of a weight change in some edge, and let ϕ be the token corresponding to the event ϵ. We say that a node u knows ϵ at time t if ϕ was used to update $A(u)$ in step 3(c) of protocol Bin at some time $t' \le t$. We say that a bin b *affects* bin b' if, by filling b sufficiently many times, b' becomes full at least once, that is, if there exists a sequence $b^{-1}, b^0, b^1, b^2, \ldots, b^k$ such that $b^{-1} = b$, $b^k = b'$, and for $0 \le i \le k$, b^i is the supervisor bin of b^{i-1}.

Let $P = (u_1, u_2, \ldots, u_k)$ be a simple path connecting u and v where $u_1 = u$ and $u_k = v$. Let $l = \lceil \log k \rceil$ (i.e., $2^{l-1} < k \le 2^l$). We regard the nodes on P as nodes of $T'(G)$. For every node $u_i \in P$, let R_i be the path from u_i to r' and let I_i be the subpath of R_i of length 2^l containing u_i. Let $I(P) = \bigcup_i I_i$. A *P-relevant* event is a weight-change event occurring in an edge of P at some time before time t. A *P-relevant token* is a token that corresponds to a P-relevant event.

LEMMA 3.4. *Let P be a simple path connecting u and v in G and let $\tilde{P}$ be the weight of P in the graph* DATA(u)*. Let ϵ be a P-relevant event and let ϕ be the token corresponding to ϵ. Let $u_i \in P$ be the node that created the token ϕ, that is, the node responsible for the edge where ϵ occurred. At time t the following holds.*

(1) *Either u knows ϵ or ϕ lies in a nonempty bin in I_i.*
(2) *The total number of P-relevant tokens that are stuck in $I(P)$ is at most $\frac{|I(P)|\delta}{3D}$.*
(3) *$\tilde{P}$ is a $\sqrt{\beta}$-approximation to $\omega(P)$, namely, $\tilde{P}/\sqrt{\beta} \le \omega(P) \le \sqrt{\beta} \cdot \tilde{P}$.*
(4) *$\tilde{P}$ is an α-approximation to $\omega(P)$.*

PROOF. If the token ϕ is not stuck in any bin in I_i, then, by Claim 3.3, ϕ was in b_i, a bin of I_i of level at least $l-4$ that became full while ϕ was in it. Therefore, by step 3(c) of protocol Bin, all vertices of (unweighted) distance 2^{l+1} from b_i in G must know of the event ϵ, and in particular u must know the event ϵ. The first part of the lemma follows.

As for the second part of the lemma, fix a level j. Note that if $b_1 \ne b_2$ are two level-j bins that are affected by two level-$j-1$ bins a_1 and a_2, respectively, then the path from a_1 to b_1 is disjoint from the path from a_2 to b_2. Each such path is of length at least 2^{j-1}; therefore the number of j-level bins in $I(P)$ affected by local bins in the u_i's is at most $|I(P)|/2^{j-1}$. Even if all these bins are full, the total number of tokens stuck in these bins is at most

$$\frac{|I(P)|}{2^{j-1}} \cdot \frac{2^j \delta}{6D(\log n + 2)} = \frac{|I(P)|\delta}{3D(\log n + 2)}.$$

Noting that there are at most $\log n + 2$ levels, the total number of P-relevant tokens that are stuck in $I(P)$ is at most $\frac{|I(P)|\delta}{3D}$ and the second part of the lemma follows.

Since each token accounts for a weight change of 1, $|\tilde{P} - \omega(P)|$ is bounded from above by the number of P-relevant tokens that u doesn't know of. Therefore, by the first two parts of this lemma, we get $|\tilde{P} - \omega(P)| \leq \frac{|I(P)|\delta}{3D}$. Since G has local density at most D and all vertices of $I(P)$ are at (unweighted) distance of at most $3k$ from u in G, we get $|I(P)| \leq D \cdot 3k$. Therefore $|\tilde{P} - \omega(P)| \leq k \cdot \delta$. The third and fourth parts of the lemma follow, since both $\tilde{P}$ and $\omega(P)$ are always at least k and since $\delta \leq \min\{\alpha - 1, \sqrt{\beta} - 1\}$. □

LEMMA 3.5. *At a quiet time t, the path $P = \mathcal{R}_{SD}(\beta)(Data(u), id(v))$ satisfies $d^{\omega}(u, v, t) \leq \omega(P) \leq \beta \cdot d^{\omega}(u, v, t)$.*

PROOF. Let P' be a simple shortest path connecting u and v such that at time t, $\omega(P') = d^{\omega}(u, v, t)$. By the third part of Lemma 3.4, $\omega(P) \leq \sqrt{\beta} \cdot \tilde{P}$ and $\tilde{P}' \leq \sqrt{\beta} \cdot \omega(P')$. Since $P = \mathcal{R}_{SD}(\beta)(Data(u), id(v))$, we have $\tilde{P} \leq \tilde{P}'$ and since $\omega(P') = d^{\omega}(u, v, t)$, we have $\omega(P') \leq \omega(P)$. Therefore $d^{\omega}(u, v, t) = \omega(P') \leq \omega(P) \leq \sqrt{\beta} \cdot \tilde{P} \leq \sqrt{\beta} \cdot \tilde{P}' \leq \beta \cdot \omega(P') = \beta \cdot d^{\omega}(u, v, t)$. The lemma follows. □

3.4.2. *Time Complexity.*

LEMMA 3.6. $\text{AVG\_TIME}(\pi_{SD}(\beta)) = O(D \log^2 n)$.

PROOF. Fix a level l. Let $b_1, b_2, \ldots$ be the level-l bins of $T'(G)$. For a level-l bin b_i, let L_i be the nodes holding the local bins that affect b_i. Note that $L_i \cap L_j = \emptyset$ for $i \neq j$. Let $E_{L_i} = \{e \in E \mid u \text{ is responsible for } e \text{ and } u \in L_i\}$. Let m_{L_i} be the number of topological changes occurring in edges of E_{L_i} during the execution and let $m = \sum_i m_{L_i}$ be the total number of changes. The number of times b_i becomes full is at most $m_{L_i}/Cap(l)$. Each time a level-l bin becomes full, its contents are broadcast to a distance of $O(2^l)$, which takes $O(2^l)$ time units. Therefore, the total time units caused by all the level-l bins is bounded by $\sum_i \frac{m_{L_i}}{Cap(l)} \cdot O(2^l) = \frac{m}{Cap(l)} \cdot O(2^l) = O(m \cdot D \cdot \log n)$. Since there are $O(\log n)$ levels, we obtain that the average time complexity per topological change is $O(D \log^2 n)$. □

Combining the preceding lemmas together with Lemma 3.2, we obtain the following theorem.

THEOREM 3.7. *Let $\beta > 1$ be constant. $\pi_{SD} = \langle \mathcal{U}_{SD}, \mathcal{R}_{SD} \rangle$ is a β-approximate source directed routing scheme for the family $\mathcal{F}(n, D)$ with the following complexities.*

(1) $\text{AVG\_TIME}(\pi_{SD}(\beta)) = O(D \log^2 n)$,

(2) $\text{DATA}(\pi_{SD}(\beta)) = O(n \cdot \log n)$.

3.5. LOWER BOUNDS. In this section we give some simple lower bounds for the database size and maximum time complexity of β-approximate source-directed routing schemes.

We first establish a lower bound for the maximum database size of β-approximate source-directed routing schemes. Note that any graph with maximum degree greater than 2 has local density greater at least 2.

LEMMA 3.8. *For any $D \geq 2$, any β-approximate source-directed routing scheme $\pi_{SD} = (\mathcal{U}_{SD}, \mathcal{R}_{SD})$ for the family $\mathcal{F}(n, D)$ has $\text{DATA}(\pi_{SD}) = \Omega(n)$.*

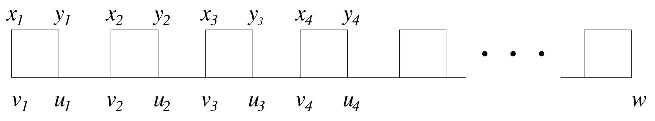

FIG. 1. The graph G.

PROOF. Consider the graph G shown in Figure 1. Note that G has local density 2 and hence G belongs to the family $\mathcal{F}(n, D)$ for any $D \geq 2$. Assume without loss of generality that the identity of the bottom rightmost vertex, w, in G is 1. Given a weight assignment to the edges, a *good* path is a (not necessarily simple) path connecting v_1 and w whose weighted length is a β-approximation to $d_G^w(v_1, w)$. Upon receiving $id(w) = 1$, $\mathcal{R}_{SD}(v_1)$ outputs a good path.

It is easy to show that for every $n/2$-bit binary string B, there exists a weight assignment ω_B to the edges of G such that (v_i, u_i) is included in every good path iff the ith bit in B is 1. Therefore $\mathcal{R}_{SD}(v_1)$ must distinguish between $2^{n/2}$ possibilities and hence $\text{DATA}(\pi_{SD}) = \Omega(n)$. □

We now show that even on underlying graphs with constant local density, the maximum time complexity on any β-approximate source-directed routing scheme is $\Omega(Diam)$. Let us note that a similar proof establishes the same result also for β-approximate point-to-point routing schemes.

LEMMA 3.9. *For any $D \geq 1$, any β-approximate source-directed routing scheme $\pi_{SD} = (\mathcal{U}_{SD}, \mathcal{R}_{SD})$ for the family $\mathcal{F}(n, D)$ has* $\text{MAX\_TIME}(\pi_{SD}) = \Omega(Diam)$.

PROOF. Let C be an n-node circle. Initially, assume all edges have weight 1. Let u and v be two vertices of distance $\Theta(n)$ on the circle. Let P be the path chosen by the router from u to v. Now, repeatedly raise the weight of the middle edge in P. At some point, P is no longer a good route from u to v, that is, $|P| \geq \beta \cdot d_C(u, v)$. Therefore, after some topological event, a message must reach u in order to let the router choose a different path from u to v. To complete this update, $\Omega(Diam)$ time units are required. □

4. *The Point-to-Point Routing Scheme $\pi_{PTP}(\beta)$*

Let $\beta > 1$ be constant. In this section we introduce our β-approximate point-to-point routing scheme $\pi_{PTP}(\beta)$ for the family $\mathcal{F}(n, D)$.

4.1. THE UPDATE PROTOCOL $\mathcal{U}_{PTP}(\beta)$. The memory structure *Memory*(v) of each vertex v is $\langle A(v), A_{int}, L(v)\rangle$. Specifically, $A(v)$ is the memory structure given to v by the source-directed update protocol $\mathcal{U}_{SD}$, and A_{int} is the initial matrix $A(v)$ representing the initial graph. The component $L(v)$ of *Memory*(v) is a table containing elements of the form $\langle \phi, b\rangle$, where b is a bin and ϕ a token.

The update protocol $\mathcal{U}_{PTP}$ uses protocol Bin' instead of protocol Bin. Protocol Bin' is the same as protocol Bin, except that in step 3 we add one more substep 3(d), described next.

Substep 3(d) of Bin'. Each node $z \in Q(b)$ adds (ϕ, b) to $L(z)$.

4.2. THE ROUTER PROTOCOL $\mathcal{R}_{PTP}(\beta)$ AND THE DATA STRUCTURE DATA(u). Recall that $A'(v)$ is the graph obtained by $A(v)$, where the weight of an edge in $A'(v)$ is $\omega(e, v)$, the first field in e's entry of $A(v)$. Let M be a message originated at

x and destined for y. The sender x initially calculates $h = d^{\omega}_{A'(x)}(x, y)$, the weighted distance between x and y in $A'(x)$. Let $q = \min\{\beta\alpha h, n\}$ and let $l = \lceil \log q \rceil$ (i.e., $2^{l-1} < q \leq 2^l$). The sender then attaches the header $H = (id(y), l)$ to the message M. Since H can be encoded with at most $2 \log n$ bits, we obtain that the header size $\mathcal{HD}(\pi_{PTP})$ is $O(\log n)$.

Denote the subgraph of G induced by all vertices whose unweighted distance to y is at most 2^l by $Ball(H) = \Gamma(y, 2^l)$. Consider the vertices of $Ball(H)$ as vertices of $T(G)$ and for every node $u_i \in Ball(H)$, let R_i be the path from u_i to r' and let I_i be the subpath of R_i, of length 2^l, containing u_i. Let $I(Ball(H)) = \bigcup_i I_i$. A bin b is $I(Ball(H))$-*universal* if $sup(b)$ is outside of $I(Ball(H))$.

In an intermediate node u along the route (including the sender x), algorithm $\mathcal{R}_{PTP}(\beta)$ operates as follows. Upon receiving H and $Memory(u)$ as input, $\mathcal{R}_{PTP}(\beta)$ creates the following edge-weight matrix $C(u, H)$ for $Ball(H)$. Initially, $C(u, H)$ is A_{int} restricted to $Ball(H)$. Now u updates $C(u, H)$ by inspecting its table $L(u)$. Let $\phi = \langle id(e), \omega(e), c\rangle$ be a token in some element $(\phi, b) \in L(u)$. If b is an $I(Ball(H))$-universal bin, ϕ corresponds to an event happening in $Ball(H)$, and ϕ is fresh with respect to $C(u, H)$, then $C(u, H)$ updates e's entry to be $(\omega(e), c)$.

After calculating $C(u, H)$, using an algorithm similar to the Dijkstra or the Bellman-Ford algorithms on the graph $C(u, H)$, the router $\mathcal{R}_{PTP}(\beta)$ efficiently calculates and outputs a port number. This port number connects u to the next vertex on a simple path P (contained in $Ball(H)$) connecting u and y, which satisfies $\tilde{P} = d^{\omega}_{C(u,H)}(u, y)$.

The database size can be reduced further. For an intermediate node u along the route from x to y, the port on which the message is to be delivered depends only on $Memory(u)$, l, and the destination y. We let $\text{DATA}(u)$ contain a table D_l for each integer $l \leq \log n$. Each such table contains n entries corresponding to the vertices of G. The y's entry in D_l contains $\mathcal{R}_{PTP}(Memory(u), id(y), 2^l)$. Given $\text{DATA}(u)$ and $H = (id(y), l)$, the router $\mathcal{R}_{PTP}$ outputs the port in y's entry in D_l. We therefore get the following lemma.

LEMMA 4.1. $\text{DATA}(\pi_{PTP}(\beta)) = O(n \log^2 n)$.

4.3. ANALYSIS.

4.3.1. *Approximation and Correctness.*

For simplicity of presentation, we first assume the route to start at some quiet time t and that the system remains quiet during the route. We then show correctness also in the case where the route starts at a nonquiet time.

Fix a quiet time t. A $Ball(H)$-*relevant* event is a weight-change event occurring in an edge e of $Ball(H)$ at some time before t. A token is $Ball(H)$-*relevant* if it corresponds to a $Ball(H)$-relevant event.

LEMMA 4.2. *Let b be an $I(Ball(H))$-universal bin. If some $Ball(H)$-relevant token ϕ was in b and b became full at time $t_0 \leq t$, then at time t, for any vertex $u \in Ball(H)$, $\langle\phi, b\rangle \in L(u)$.*

PROOF. Let k be the level of b and let $u \in Ball(H)$. Let ϵ be the event corresponding to ϕ and let $e \in Ball(H)$ be the edge where ϵ occurred. Let $u' \in Ball(H)$ be the node responsible for e. Since $sup(b)$ is outside of $I(Ball(H))$, we get that $d_T(sup(b), u') \geq 2^l$. Hence, by Claim 3.3, $k \geq l - 4$. On the other hand,

$d_G(u', b) \leq 3 \cdot 2^{k+1} \leq 6 \cdot 2^k$. Therefore $d_G(u, b) \leq d_G(u, u') + d_G(u', b) \leq 2^l + 6 \cdot 2^k \leq 2^4 \cdot 2^k + 6 \cdot 2^k \leq 2^{5+k}$, yielding $u \in Q(b)$. The lemma follows by step 3(d) of protocol Bin' and by the fact that $u \in Q(b)$. □

CLAIM 4.3. *For any two vertices $w, z \in Ball(H)$ on the route $P(x, y)$, at any time during the routing, $C(w, H) = C(z, H)$.*

PROOF. Let ϕ be a token that was used to update $C(z, H)$ during the route. Since the system remains quiet during the route, by Lemma 4.2, ϕ was used to update $C(w, H)$ as well. The claim follows. □

LEMMA 4.4. *The weighted length of the resulting route $P(x, y)$ between x and y is a β-approximation of $d^{\omega}_G(x, y, t)$.*

PROOF. For a path P in $Ball(H)$, let $\omega_C(P)$ be the weighted length of P in $C(x, H)$. Let P' be a path minimizing the distance between x and y in G, namely, at time t, $\omega(P') = d^{\omega}_G(x, y, t)$. Note that by Lemma 3.4, P' is contained in $Ball(H)$. Let P^* be a path from x to y contained in $Ball(H)$ and satisfying $\omega_C(P^*) = d^{\omega}_{C(x,H)}(x, y)$. Consider some $Ball(H)$-relevant token ϕ. By Lemma 4.2 we get that at time t, the only $Ball(H)$-relevant tokens that were not used while updating $C(x, h)$ are those stuck in bins of $I(Ball(H))$. Since each token accounts for a weight change of at most 1, by the second part of Lemma 3.4, we get that for every path $P \in Ball(H)$,

$$|\omega_C(P) - \omega(P)| \leq \frac{|I(P)|\delta}{3D} \leq \frac{|I(Ball(H))|\delta}{3D} \leq \frac{|I(Ball(H))|(\alpha - 1)}{3D\alpha\beta}.$$

Note that all vertices of $I(Ball(H))$ are of unweighted distance (in G) of at most $3\alpha\beta h$. As G has local density at most D, $|I(Ball(H))| \leq 3\beta\alpha D h$. Therefore $|\omega_C(P) - \omega(P)| \leq h(\alpha - 1)$. By the fourth part of Lemma 3.4, h is an α-approximation to $d^{\omega}_G(u, v, t) = \omega(P')$, hence for all $P \in Ball(H)$, $|\omega_C(P) - \omega(P)| \leq (\alpha^2 - \alpha)\omega(P')$. In particular, $|\omega_C(P') - \omega(P')| \leq (\alpha^2 - \alpha)\omega(P')$. Therefore $\omega_C(P') \leq (\alpha^2 - \alpha + 1)\omega(P') = \frac{\beta+1}{2} \cdot \omega(P')$. For the same reasoning we get $|\omega(P^*) - \omega_C(P^*)| \leq (\alpha^2 - \alpha)\omega(P')$. Therefore $\omega(P^*) \leq (\alpha^2 - \alpha)\omega(P') + \omega_C(P^*) \leq (\alpha^2 - \alpha)\omega(P') + \omega_C(P') \leq (\alpha^2 - \alpha + \frac{\beta+1}{2}) \cdot \omega(P') \leq \beta\omega(P')$. If follows that $\omega(P') \leq \omega(P^*) \leq \beta\omega(P')$. By Claim 4.3, for any vertex $z \in Ball(H)$, $C(z, H) = C(x, H)$. Therefore, the resulting path P_{route} must satisfy $\omega_C(P_{route}) = d^{\omega}_{C(x,H)}(x, y)$, hence $\omega(P') \leq \omega(P_{route}) \leq \beta\omega(P')$ and the lemma follows. □

Let us now prove correctness also in the case where the route from x to y does not necessarily start at a quiet time. Note that also in this case, all those vertices participating in the route are in $Ball(H)$. If at some quiet time t the message is at u and the system remains quiet during the remaining route, then by Lemmas 4.2 and 4.3, for every two vertices w and z in the remaining route from u to y, $C(w, H) = C(z, H)$. Therefore, the path resulting from this remaining route is a path (contained in $Ball(H)$) connecting u and y. In particular, note that the message reaches y. Since we assume the system is eventually quiet for a sufficiently long time period, the correctness of the scheme follows.

4.3.2. *Complexities.*

LEMMA 4.5. $\text{AVG\_TIME}(\pi_{PTP}(\beta)) = O(D\log^2 n)$.

PROOF. The lemma follows from Lemma 3.6 and from the fact that the messages sent by protocol Bin' are the same as those sent by protocol Bin. □

By combining Lemmas 4.1, 4.4, and 4.5, we obtain the following theorem.

THEOREM 4.6. *Let $\beta > 1$ be constant. Then $\pi_{PTP}(\beta) = \langle \mathcal{U}_{PTP}, \mathcal{R}_{PTP} \rangle$ is a β-approximate point-to-point routing scheme for the family $\mathcal{F}(n, D)$ with the following complexities.*

(1) $\text{AVG\_TIME}(\pi_{PTP}(\beta)) = O(D\log^2 n)$.

(2) $\text{DATA}(\pi_{PTP}(\beta)) = O(n \cdot \log^2 n)$.

(3) $\mathcal{HD}(\pi_{PTP}) = O(\log n)$.

REFERENCES

AFEK, Y., AWERBUCH, B., PLOTKIN, S. A., AND SAKS, M. 1996. Local management of a global resource in a communication network. *J. ACM 43*, 1–19.

AFEK, Y., GAFNI, E., AND RICKLIN, M. 1989. Upper and lower bounds for routing schemes in dynamic networks. In *Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS)*, 370–375.

AWERBUCH, B., BAR-NOY, A., LINIAL, N., AND PELEG, D. 1990. Improved routing strategies with succinct tables. *J. Algor. 11*, 307–341.

AWERBUCH, B., AND PELEG, D. 1992. Routing with polynomial communication-space trade-off. *SIAM J. Discr. Math. 5*, 307–341.

CHINN, P., CHAVATÁLOVÁ, J., DEWDNEY, A., AND GIBBS, N. 1982. The bandwidth problem for graphs and matrices—Survey. *J. Graph Theory 6*, 223–254.

COWEN, L. 2001. Compact routing with minimum stretch. *J. Algor. 38*, 170–183.

CHUNG, R. F., AND SEYMOUR, P. D. 1989. Graphs with small bandwidth and cutwidth. *Discr. Math. 75*, 113–119.

DOLEV, S., KRANAKIS, E., KRIZANC, D., AND PELEG, D. 1999. Bubbles: Adaptive routing scheme for high-speed dynamic networks. *SIAM J. Comput. 29*, 804–833.

EPPSTEIN, D., GALIL, Z., AND ITALIANO, G. F. 1999. Dynamic graph algorithms. In *Algorithms and Theoretical Computing Handbook*, Atallah, M. J. CRC Press, Boca Raton, FL, Chapter 8.

FRAIGNIAUD, P., AND GAVOILLE, C. 1997. Universal routing schemes. *Distrib. Comput. 10*, 65–78.

FEIGE, U. 2000. Approximating the bandwidth via volume respecting embeddings. *J. Comput. Syst. Sci. 60*, 510–539.

FEIGE, U., AND TALWAR, K. 2005. Approximating the bandwidth of caterpillars. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*. Lecture Notes in Computer Science, vol. 3624. Springer, 62–73.

FEIGENBAUM, J., AND KANNAN, S. 2000. Dynamic graph algorithms. In *Handbook of Discrete and Combinatorial Mathematics*, CRC Press, Boca Raton, FL.

IWAMA, K., AND KAWACHI, A. 2000. Compact routing with stretch factor less than three. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC)*, 337.

IWAMA, K., AND OKITA, M. 2003. Compact routing for flat networks. In *Proceedings of the 17th International Symposium on Distributed Computing*.

KRAUTHGAMER, R., LEE, J., MENDEL, M., AND NAOR, A. 2004. Measured descent: A new embedding method for finite metrices. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*.

KORMAN, A. 2005. General compact labeling schemes for dynamic trees. In *Proceedings of the 19th International Symposium on Distributed Computing*.

KORMAN, A., PELEG, D., AND RODEH, Y. 2004. Labeling schemes for dynamic tree networks. *Theory Comput. Syst. 37*, 49–75.

KORMAN, A., AND PELEG, D. 2003. Labeling schemes for weighted dynamic trees. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP)*, 369–383.

LINIAL, N. 1992. Locality in distributed graph algorithms. *SIAM J. Comput. 21*, 193–201.
PELEG, D. 2000. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, Philadelphia, PA.
PELEG, D., AND UPFAL, E. 1989. A tradeoff between size and efficiency for routing tables. *J. ACM 36*, 510–530.
SANTORO, N., AND KHATIB, R. 1985. Labeling and implicit routing in networks. *The Comput. J. 28*, 5–8.
THORUP, M., AND ZWICK, U. 2001. Compact routing schemes. In *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 1–10.

RECEIVED SEPTEMBER 2006; ACCEPTED MARCH 2007