Low Complexity Variants of the Arrow Distributed Directory

David Peleg and Eilon Reshef

Department of Computer Science and Applied Mathematics, The Weizmann Institute, Rehovot 76100, Israel
E-mail: peleg@wisdom.weizmann.ac.il

Received January 25, 2000; revised July 12, 2001

This paper considers an enhancement to the arrow distributed directory protocol, introduced in [8]. The arrow protocol implements a directory service, allowing nodes to locate mobile objects in a distributed system, while ensuring mutual exclusion in the presence of concurrent requests. The arrow protocol makes use of a minimum spanning tree $T_m$ of the network, selected during system initialization, resulting in a worst-case overhead ratio of $(1 + \text{stretch}(T_m))/2$. However, we observe that the arrow protocol is correct communicating over any spanning tree $T$ of $G$. We show that the worst-case overhead ratio is minimized by the minimum stretch spanning tree and that the problem cannot be approximated within a factor better than $(1 + \sqrt{5})/2$, unless $P = \mathcal{NP}$. In contrast, other trees may be more suitable if one is interested in the average-case behavior of the network. We show that in the case where the distribution of the requests is fixed and known in advance, the expected communication is minimized using the minimum communication cost spanning tree (MCT). It is shown that the resulting MCT problem is a restricted case for which one can find a tree $T$ over which the communication cost of the arrow protocol is at most 1.5 times the expected communication cost of an optimal protocol. We also show that even if the distribution of the requests is not fixed, or not known to the protocol in advance, then if the adversary is oblivious, one may use probabilistic approximation of metric space [2, 3] to ensure an expected overhead ratio of $O(\log n \log \log n)$ in general and an expected ratio of $O(\log n)$ in the case of constant dimension Euclidean graphs.

© 2001 Elsevier Science (USA)

1. INTRODUCTION

Many distributed systems support some concept of mobile objects. A mobile object can be a file, a process, or any other data structure. Such an object lives...
on one node at a time and moves from one node to another in response to requests by clients. A directory service allows nodes to locate mobile objects (navigation) and ensures mutual exclusion in the presence of concurrent requests (synchronization). In such a system, a directory service allows nodes to locate such objects, while ensuring mutual exclusion in the presence of concurrent requests.

The arrow distributed directory protocol [8] is a simple and elegant protocol for implementing such a directory service. To formally analyze the behavior of the arrow distributed directory protocol, we model the distributed system as a weighted graph $G = (V, E, \omega)$, where each vertex represents a node in the system, and each edge represents a bidirectional communication link. The cost of sending a message along an edge $e$ is the weight of the edge $\omega(e)$.

The arrow distributed directory protocol implements a directory that tracks the location of a simple object by a spanning tree $T$ of $G$ that serves as the communication backbone of the directory service. Whereas the original presentation of the protocol [8] refers to a minimum-weight spanning tree (MST), we observe that the arrow protocol is correct communicating over any spanning tree $T$ of $G$, and thus, one may consider using trees other than an MST as the communication backbone of the protocol.

The protocol operates as follows. Each node $v$ holds a single pointer, denoted $\text{link}(v)$, which may point to any of its neighbors in the tree $T$. If $\text{link}(v)$ is not null, the mobile object is expected to be in the tree component containing $\text{link}(v)$. When a node $r$ requests the object, it sends a find message to $\text{link}(r)$ and clears $\text{link}(r)$. When a node $u$ receives a find message from a neighbor $v$ in the tree, it first examines $\text{link}(u)$. If $\text{link}(u)$ is $w$, then $u$ forwards the message to $w$. Otherwise, the object is in $u$ or is expected to arrive at $u$, and thus $u$ buffers the request (until it receives the object and has completed using it) and sets $\text{link}(u)$ to point to $v$. When the object is ready to be sent, $u$ sends the object to the requester $r$ along the shortest path in the graph $G$ between $u$ and $r$. The directory tree is initialized so that following the links from any node leads to the node where the object resides. Despite the protocol’s simplicity, it is shown in [8] that the protocol ensures mutually exclusive access to the object and ensures that every node that sends a find request eventually receives the object.

To analyze the overhead imposed by the arrow protocol $A$, the communication cost of the protocol is compared against an optimal directory $OPT$ in which synchronization and navigation are “free.” The optimal directory accepts only serial requests (namely, concurrent requests are disallowed) and delivers each request directly. This implies, in particular, that both the optimal directory and our algorithm service the requests in the same order, namely, the optimal directory is not allowed to anticipate future requests and delay servicing the current one for improving performance. The same assumption is made also in the analysis of the original arrow directory protocol [8].

Consider a sequence $S$ of requests issued by the nodes $v_{k_1}, ..., v_{k_L}$. To deliver the object from $v_{k_i}$ to $v_{k_{i+1}}$, the arrow directory routes a find message from $v_{k_{i+1}}$ to $v_{k_i}$. 
along the unique path in the spanning tree $T$, paying $d_T(v_{k_i+1}, v_{k_i})$, where $d_T(v_i, v_j)$ is the weight of the path in $T$ between $v_i$ and $v_j$, and routes the object back from $v_{k_i+1}$ along the shortest path in the graph, paying another $d_G(v_k, v_{k+1})$, where $d_G(v_i, v_j)$ is the weight of the shortest path in $G$ between $v_i$ and $v_j$. In contrast, the optimal directory routes both the request and the object along the shortest path, paying $2d_G(v_k, v_{k+1})$.

Denote
\[ c(S, T) \triangleq \sum_{i=1}^{l-1} d_T(v_{ki}, v_{ki+1}), \quad c(S, G) \triangleq \sum_{i=1}^{l-1} d_G(v_{ki}, v_{ki+1}). \]

The overall communication of the arrow protocol serving the sequence $S$ over the tree $T$ is
\[ c(S, A) = c(S, T) + c(S, G). \]

whereas the overall communication cost of the optimal directory is
\[ c(S, OPT) = 2c(S, G). \]

Thus, the overhead ratio $\rho$ imposed by the arrow protocol on a sequence of requests $S$ is
\[ \rho(S) \triangleq \frac{c(S, A)}{c(S, OPT)} = \left(1 + \frac{c(S, T)}{c(S, G)}\right)/2. \]

We discuss two measures of the overhead ratio, namely, the worst-case overhead and the average-case overhead. The worst-case overhead is the maximal value of $\rho(S)$, taken over all possible serial executions $S = \langle v_{k_1}, ..., v_{k_l} \rangle$, i.e.,
\[ \text{OH} \triangleq \max_S \{\rho(S)\}. \]

To minimize $\text{OH}$, one is required to find a tree $T$ that minimizes the overhead of a single request, i.e., minimizes
\[ \text{stretch}(T) \triangleq \max_{i, j} \left\{ \frac{d_T(v_i, v_j)}{d_G(v_i, v_j)} \right\}. \]

This problem, called minimum stretch spanning tree (MSST), is known to be $NP$-hard [5], and it is shown in Section 2 that the problem cannot be approximated better than $(1 + \sqrt{5})/2$, unless $NP = P$. It is thus reasonable to develop tree construction algorithms for optimizing the average-case behavior of the arrow protocol.

In analyzing the average-case behavior, we are interested in the expected communication cost $E[c(S, A)]$, where the expectation is taken over all possible serial executions of length $\ell$ and over the possible coin tosses of the protocol $A$. In addition, we may look at the expected overhead, which is the expected value of $\rho(S)$,
\[ \overline{OH} \triangleq E[\rho(S)]. \]
In these cases, we show that trees other than the minimum stretch spanning tree may ensure lower communication cost.

We consider two different models of the network behavior. In Section 3, we discuss the independent (IND) model. In this model, a probability $p_i$ is associated with each node $v_i$ and requests to the mobile object are generated independently according to the probability distribution $\hat{P} = (p_1, \ldots, p_n)$; i.e., the probability that the next request is generated by $v_i$ is $p_i$, independent of previous requests. We show that in the IND model, the optimal tree is the solution for a special instance of the minimum communication spanning tree (MCT) problem defined next, based on the probability distribution $\hat{P}$.

The MCT problem was introduced in [10] and can be formalized as follows. An instance of the problem is a complete undirected graph $G = (V, E)$, where every pair of vertices $\{v_i, v_j\}$ in $V$ is assigned a nonnegative weight $w(v_i, v_j)$ and a non-negative communication requirement $r(v_i, v_j)$. We are asked to find a spanning tree $T$ of $G$ for which the total communication cost, i.e.,

$$c_{\text{MCT}}(T) \triangleq \sum_{i,j} [r(v_i, v_j) \cdot d_T(v_i, v_j)],$$

is minimized. MCT is known to be $NP$-hard even in the uniform case, in which all requirements are equal; i.e., $r(v_i, v_j) = 1$ for every $i$ and $j$ [11].

We show that the MCT problem that corresponds to the IND probabilistic model of the network behavior is a restricted case in which the requirement matrix is a product of the vector $\hat{P}$ and its transpose, i.e., $R = \hat{P}^T \cdot \hat{P}$; hence $r(v_i, v_j) = p_i \cdot p_j$. We refer to this special case of the MCT problem as the independent-requirements (IR) MCT problem. Furthermore, we show that in this special case one can find in polynomial time a spanning tree $T$ whose communication cost is at most twice the communication cost in the original graph $G$. This implies that the expected communication cost of the arrow protocol over $T$ is at most 1.5 times the expected communication cost of the optimal directory protocol; i.e., $\mathbb{E}[c(S, A)] \leq 1.5 \cdot \mathbb{E}[c(S, OPT)]$. We show that this bound is tight.

In Section 4, we consider the oblivious (OBLIV) model, in which the distribution of the requests is not fixed, or not known to the protocol in advance. In this case, we show that if the adversary is oblivious [4], in the sense that it does not see the coin tosses of the tree construction algorithm or it specifies the series of requests $v_k, \ldots, v_n$ in advance, one may use a randomized tree construction algorithm based on probabilistic approximation of metric spaces [2, 3]. This construction ensures an expected overhead ratio of $OH = O(\log n \log \log n)$ in general and an expected overhead ratio of $OH = O(\log n)$ in the case of constant dimension Euclidean graphs (relying on [7, 14]).

In the rest of the discussion, we assume that the graph at hand is complete and metric; i.e., the weights obey the triangle inequality. Note that if this is not the case, one may complete the graph into a complete metric graph by adding “virtual” links, physically implemented by the corresponding shortest paths, without affecting the results.
2. MINIMUM STRETCH SPANNING TREE

Given a spanning tree \( T \) over a weighted graph \( G = (V, E, \omega) \), let \( \text{stretch}(T) \) denote the maximum stretch over all pairs of vertices in the tree \( T \), i.e.,

\[
\text{stretch}(T) \triangleq \max_{i,j} \{ \text{stretch}_T(v_i, v_j) \},
\]

where \( \text{stretch}_T(v_i, v_j) \triangleq d_T(v_i, v_j)/d_G(v_i, v_j) \). The MSST problem is to find the spanning tree \( T \) that minimizes \( \text{stretch}(T) \). The problem was shown to be \( \mathcal{NP} \)-hard in [5].

It is interesting to note that although there are algorithms that ensure an average stretch of \( O(\log n \log \log n) \) [3], no simple upper bound holds in the case of the worst-case stretch. For example, any spanning tree of the \( n \)-vertex ring has an average stretch slightly less than 2, but a worst-case stretch of \( n-1 \). Further, in the case of the 2-dimensional \( n \)-vertex grid, the results of [1] imply the existence of a spanning tree with an average stretch of \( O(\log n) \), but also that any spanning tree has a pair of vertices whose stretch is \( \Omega(\sqrt{n}) \).

The following lemma provides an alternative, and sometimes more convenient, characterization of \( \text{stretch}(T) \).

**Lemma 2.1.** \( \text{stretch}(T) = \max_{(v_i, v_j) \in E} \{ \text{stretch}_T(v_i, v_j) \} \).

**Proof.** Let \( s = \max_{(v_i, v_j) \in E} \{ \text{stretch}_T(v_i, v_j) \} \). Clearly \( s \leq \text{stretch}(T) \). For the converse, consider any two vertices \( v_i \) and \( v_j \), and let \( \langle v_{i_1} = v_i, v_{i_2}, \ldots, v_{i_k} = v_j \rangle \) be some shortest path between them. Then,

\[
d_T(v_i, v_j) \leq \sum_{i=1}^{k} d_T(v_{i-1}, v_i) \leq s \cdot \sum_{i=1}^{k} d_G(v_{i-1}, v_i) = s \cdot d_G(v_i, v_j).
\]

Note that the same argument shows that one can also ignore pairs \( (v_i, v_j) \) for which \( \omega(v_i, v_j) > d_G(v_i, v_j) \).

In the remainder of this section we show that MSST cannot be approximated within any ratio \( \rho < (1 + \sqrt{5})/2 \). We give a reduction from the 3SAT problem defined as follows. An instance \( I \) of 3SAT is a set of Boolean variables \( x_1, \ldots, x_n \) and a set of disjunctive clauses \( c_1, \ldots, c_m \), each containing exactly three literals, where a literal is either a variable \( x_i \) or its negation. The satisfiability problem is to decide whether there exists an assignment to the variables \( x_i \) such that all the clauses are satisfied.

Given an instance \( I \) of 3SAT, we construct an instance \( G_I = (V_I, E_I) \) of MSST with a gap of \( (1 + \sqrt{5})/2 - \varepsilon \); namely, if a satisfying assignment exists, then the graph contains a spanning tree \( T \) whose stretch is at most \( t \), for some \( t \). Otherwise, every spanning tree \( T \) of \( G_I \) has a stretch of \( ((1 + \sqrt{5})/2 - \varepsilon) \cdot t \) or more.
Theorem 2.2. For every $\varepsilon > 0$, no polynomial time algorithm can approximate the minimum stretch spanning tree within a factor of \((1 + \sqrt{5})/2 - \varepsilon\), unless $\mathcal{P} = \mathcal{NP}$.

Proof sketch. The proof is along the lines of [5]. Construct an instance $G = (V_I, E_I)$ of MSST as follows.

The vertex set $V_I$ contains a vertex for each clause $c_j$, a vertex for each literal $x_i$ or $\overline{x}_i$, and an additional vertex $z$. The edge set $E_I$ contains a path between each clause $c_j$ and each literal $x_i$ or $\overline{x}_i$ it contains, a path between each literal $x_i$ and its negation $\overline{x}_i$, and an edge between $z$ and each literal $x_i$ or $\overline{x}_i$. (See Fig. 1.)

Throughout the discussion below real numbers are used; rounding may be needed to convert the numbers into rationals. The construction uses a value $t$ which is assumed to be large enough to accommodate for the rounding. In particular, we assume $t \geq 1000/\varepsilon$.

The following gadget is used to force an edge $e = (v_i, v_j) \in E_I$ to be included in any candidate spanning tree $T$: replace the edge $e$ with a path of $2t$ edges, $e_1', \ldots, e_{2t}'$, each of weight $\omega(e)/2t$. Clearly, if all the edges $e_i'$ are included in a spanning tree $T$, their stretch is 1. Otherwise, if an edge $e_i' = (u, v)$ is not included in the spanning tree, $d_T(u, v) \geq \omega(e) \geq 2t \cdot d_G(u, v)$, and hence its stretch is at least $2t$. Edges replaced by the above gadget are called protected and appear as a double line (=) in Fig. 1.

Following are the exact details of the construction. The path connecting $z$ and a literal vertex $x_i$ or $\overline{x}_i$ consists of exactly one edge with weight $l_1 = 1$. The path connecting $x_i$ and $\overline{x}_i$ consists of a protected edge of length $l_2 = t - 1$. The path connecting a clause $c_j$ and a contained literal $x_i$ or $\overline{x}_i$ is composed of edges of weight $1 + 2\alpha + \varepsilon$, with a total path length of $l_3 = \alpha t$. The optimal value of $\alpha$ follows from the proof and is set to be $(\sqrt{5} - 1)/4$.

FIG. 1. Constructing the graph $G_I$. 
A spanning tree $T$ induces a truth assignment $\varphi$ such that $x_i$ is assigned “true” (resp. “false”) if the edge $(z, x_i)$ (resp. $(z, \bar{x}_i)$) is in $T$. To complete the proof of the theorem, we prove the following two complementary observations. If an instance $I \in 3\text{SAT}$ has a satisfying assignment, then there exists a spanning tree $T_I$ such that $\text{stretch}(T_I) \leq t$. On the other hand, if and instance $I \in 3\text{SAT}$ does not have a satisfying assignment, then $\text{stretch}(T) \geq ((1 + \sqrt{3}/2 - \varepsilon)t$ for every spanning tree $T$ of $G_I$.

Now, the following two lemmas are in place.

**Lemma 2.3.** If an instance $I \in 3\text{SAT}$ has a satisfying assignment, then there exists a spanning tree $T_I$ such that $\text{stretch}(T_I) \leq t$.

**Proof.** The spanning tree $T_I$ includes the following edges:

- The protected edges between $x_i$ and $\bar{x}_i$, for every $i$.
- A path connecting $z$ to each literal $x_i$ or $\bar{x}_i$ which is assigned “true” by the satisfying assignment.
- The paths connecting each clause $c_j$ to a literal that satisfies it (if more than one such literal exists, select one arbitrarily).
- All edges but one along the path connecting $c_j$ to any other literal it contains.

It can be easily verified that the above construction is indeed a tree. Furthermore, the stretch of the two edges $(z, x_i)$ and $(z, \bar{x}_i)$ is at most $t$. Now, since each clause $c_j$ is satisfied by at least one literal, say $x_i$, $T_I$ contains the path $c_j - x_i - z$. Thus, for the edge $u, v$ removed from the path $(c_j, x_k)$, where $x_k \neq x_i$ is a literal contained in $c_j$,

$$d_T(u, v) \leq 2 \cdot l_3 + l_1 + l_2 = (1 + 2\alpha) t + 1 < t \cdot d_G(u, v).$$

**Lemma 2.4.** If an instance $I \in 3\text{SAT}$ does not have a satisfying assignment, then for every spanning tree $T$ of $G_I$, $\text{stretch}(T) \geq ((1 + \sqrt{3}/2 - \varepsilon)t$.

**Proof.** Consider a spanning tree $T$. Assume that all the protected edges are in the spanning tree, since otherwise the lemmas holds. Now, consider the case where for some $i$ none of the two edges $(z, x_i), (z, \bar{x}_i)$ are in $T$. Then for either $x_i$ or $\bar{x}_i$ (say $x_i$),

$$d_T(z, x_i) \geq 2 \cdot l_3 + l_1 + l_2 = 2\alpha t + 1 + t - 1 = (1 + 2\alpha) t \cdot d_G(z, x_i).$$

Otherwise, for all $i$, $T$ contains exactly one of the two paths $(z, x_i)$ and $(z, \bar{x}_i)$. Define a truth assignment $\psi$ such that $\psi(x_i)$ is assigned “true” if the edge $(z, x_i)$ is in $T$ and “false” otherwise. Since by the theorem hypothesis $\psi(\cdot)$ is not a satisfying assignment, at least one clause $c_j$ is not satisfied; namely, none of the literals in $c_j$ is assigned “true.” Now, since $T$ is a spanning tree, it contains a path $(c_j, x_i)$ for some literal $x_i$ in $c_j$. The crux of the argument is that for some literal $x_i \neq x_i$ in $c_j$, at least
one edge \((u, v)\) is not contained in \(T\), as otherwise a loop would be formed and thus the distance between \(u\) and \(v\) in \(T\) is at least
\[
d_T(u, v) \geq l_1 + l_2 + 2 \cdot l_1 + l_2 + (l_3 - d_G(u, v)) = (2 + 2x) \cdot t - d_G(u, v)
\]
\[
= \left[\left(1 + \frac{1 - e}{1 + 2x + e}\right) \cdot t - 1\right] \cdot d_G(u, v)
\]
\[
> \left(1 + \frac{1 - e}{1 + 2x}\right) \cdot t \cdot d_G(u, v).
\]

The expression \(\rho = \min\{1 + 2x, 1 + \frac{1}{1 + 2x}\}\) is maximized with \(\rho = (1 + \sqrt{5})/2\) and the lemma follows.

This concludes the proof of the theorem.

\[\square\]

3. KNOWN DISTRIBUTION

In this section we discuss the average-case behavior of the arrow protocol in the IND model. This model assumes an underlying distribution based on a probability vector \(\vec{P} = (p_1, \ldots, p_n)\) s.t. \(\sum p_i = 1\), i.e., the probability that the next request is generated by \(v_i\) is \(p_i\), independent of previous requests.

It turns out that in this model, the communication cost is minimized when the communication backbone of the protocol is exactly the minimum communication cost spanning tree (MCT) for the independent-requirements case based on \(\vec{P}\).

\textbf{Theorem 3.1.} \textit{In the IND model, the expected communication cost of the arrow protocol is minimized using a tree \(T\) which is the minimum communication cost spanning tree for a network \(G\) with independent requirements \(r(v_i, v_j) = p_i \cdot p_j\).}

\textbf{Proof.} Consider a sequence of requests \(S = \langle v_{k_1}, \ldots, v_{k_\ell} \rangle\) served over a spanning tree \(T\). By linearity of expectation, the expected communication cost of the protocol over \(T\) is
\[
\mathbb{E}[c_r(S, \mathcal{A})] = \sum_{1 \leq i \leq \ell} \mathbb{E}[d_T(v_{k_i}, v_{k_{i+1}}) + d_G(v_{k_i}, v_{k_{i+1}})].
\]

Under the distribution \(\vec{P}\), and since the requests are independent, the probability that in the \(m\)th request the object is delivered from node \(v_i\) to node \(v_j\) is exactly \(p_i \cdot p_j\). Thus, the expected communication of any single request is fully determined by the probability distribution \(\vec{P}\). Denote the cost of a single request
\[
c_r(T) \triangleq \mathbb{E}[d_T(v_{k_i}, v_{k_{i+1}})] = \sum_{i,j} p_i p_j d_T(v_i, v_j)
\]
and
\[
c_r(G) \triangleq \mathbb{E}[d_G(v_{k_i}, v_{k_{i+1}})] = \sum_{i,j} p_i p_j d_G(v_i, v_j).
\]
Using this notation, $E[c, (S, T)] = (\ell - 1) \cdot c_R(T)$, $E[c, (S, G)] = (\ell - 1) \cdot c_R(G)$, and $E[c, (S, A)] = (\ell - 1) \cdot (c_R(T) + c_R(G))$.

Since $c_R(G)$ is independent of $T$, the communication cost is minimized using a tree that minimizes $c_R(T)$ (independent of the sequence $S$), i.e., minimizes $\sum_{i,j} p_i p_j d_T(v_j, v_i)$. The problem of finding such a tree is precisely the IR instance of MCT defined for $P$.

Next we present a 2-approximation algorithm IR-MCT for the independent-requirements MCT problem defined above.

**Algorithm IR-MCT.** Let $T_i$ denote a shortest-path tree from $v_i$, i.e., a tree for which $d_{T_i}(v_i, v_j) = d_G(v_i, v_j)$ for every $j$. If there is more than one such tree, select one arbitrarily.

Algorithm IR-MCT examines the trees $T_i$ for all $i, 1 \leq i \leq n$, and selects the best shortest-path tree, namely, the tree $T_i$ for which the communication cost $c_R(T_i)$ is minimized.

**Analysis.** The following lemma asserts that the cost of the best tree is at most twice the communication cost in the original graph, namely $E[c_R(T)] \leq 2c_R(G)$.

**Lemma 3.2.** There exists a vertex $v_i$ such that $c_R(T_i) \leq 2c_R(G)$.

**Proof.** Select a tree $T$ randomly from $\{T_1, ..., T_n\}$, picking $T_i$ with probability $p_i$.

The expectation of the cost of $T$ is

$$E[c_R(T)] = \sum_{i} p_i \cdot c_R(T_i) = \sum_{i} p_i \cdot \left( \sum_{j} p_j d_T(v_j, v_k) \right)$$

$$\leq \sum_{i} p_i \cdot \left( \sum_{j,k} p_j p_k (d_T(v_j, v_i) + d_T(v_i, v_k)) \right).$$

Since $T_i$ is a shortest-path tree from $v_i$, $d_T(v_i, v_j) = d_G(v_i, v_j)$ for every $j$, and thus

$$E[c_R(T)] \leq \sum_{i} p_i \cdot \left( \sum_{j,k} p_j p_k (d_G(v_j, v_i) + d_G(v_i, v_k)) \right)$$

$$= \sum_{i} p_i \cdot \left( 2 \cdot \sum_{j=1} p_j d_G(v_j, v_i) \right) = 2c_R(G).$$

Therefore, at least one of the trees $T_i$ satisfies $c_R(T_i) \leq 2c_R(G)$.

**Corollary 3.3.** In the IND model, one can find in polynomial time a tree $T$ for which the expected communication cost satisfies $E[c(S, T)] \leq 1.5 \cdot E[c(S, \mathcal{OPT})]$.

**Proof.** Run algorithm IR-MCT to yield a spanning tree $T$. By the lemma, the algorithm ensures that $c_R(T) \leq 2c_R(G)$. Therefore, $E[c, (S, \mathcal{OPT})] \leq 3(\ell - 1) c_R(G)$.

On the other hand, the optimal directory would still need to communicate back and forth along the shortest path, so the optimal communication is

$$E[c(S, \mathcal{OPT})] = 2 \cdot E[c, (S, G)] = 2(\ell - 1) c_R(G).$$

Thus, $E[c(S, \mathcal{OPT})] \leq 1.5 \cdot E[c(S, \mathcal{OPT})]$. □
Note that the result of Lemma 3.2 is essentially tight. To see this, consider the instance of the problem defined on the $n$-vertex clique $G_n$ with $p_i = 1/n$ for every $i$. For this instance, $c_R(G_n) = 1/n^2 \cdot (n \cdot 0 + n(n-1) \cdot 1) \approx 1$. On the other hand, in every shortest-path spanning tree $T_i$ of $G_n$, rooted at $v_i$, $d_{T_i}(v_i, v_j) = 1$ for every $j \neq i$, and $d_{T_i}(v_i, v_j) = 2$ for every $j \neq i \neq j$, and thus $c_R(T_i) = 1/n^2 \cdot (n \cdot 0 + (n-1) \cdot 1 + (n-1)(n-2) \cdot 2) \approx 2$. (Clearly, any other spanning tree for $G_n$ must have an even higher communication cost.)

Note also that Algorithm IR-MCT is in fact a 2-approximation algorithm for the independent-requirements MCT problem, i.e., $c_{MCT}(T) \leq 2 \cdot c_{MCT}(T_g)$, where $T_g$ is the minimum communication cost spanning tree. This result is not the best possible, though. In particular, the algorithm of [16] can be generalized to provide a polynomial time approximation scheme (PTAS) for the independent-requirements MCT problem, i.e., for any fixed $\varepsilon > 0$ it is possible to construct (in polynomial time) a tree $T_g$ such that $c_{MCT}(T_g) \leq (1 + \varepsilon) \cdot c_{MCT}(T^*)$. Unfortunately, the communication cost of $T_g$ does not seem to compare directly to $c_R(G)$, and hence this technique cannot be used to provide a provably better ratio for the purposes of the current paper.

Further, the above construction suggests that even if the behavior of the network is unknown in advance, a heuristic that randomly selects a shortest-path tree may provide better performance in practice than the minimum-weight spanning tree heuristic used in [8]. In particular, if the graph $G$ is unweighted, the shortest-path heuristic provides a nontrivial result, whereas all spanning trees $T$ of $G$ have the same weight.

4. OBLIVIOUS ADVERSARY

In this section we consider the expected average-case behavior of the arrow protocol in the OBLIV model, in which the distribution of the requests is not fixed or not known to the protocol in advance. Rather, we assume that the adversary is oblivious [4], so it does not see the coin tosses of the tree construction algorithm or specifies the series of requests $v_{k_1}, \ldots, v_{k_l}$ in advance.

We use the following definition, due to [2]:

**Definition 4.1 [2].** A complete metric graph $G$ is $\alpha$-probabilistically approximated by a probability distribution $\mathcal{D}_G$ of (spanning) trees of $G$ if for every pair of vertices $v_i, v_j$,

$$\mathbb{E}_{\mathcal{D}_G}[d_T(v_i, v_j)] \leq \alpha.$$

The following results realize the above definition.

**Proposition 4.2 [3].** Every complete metric graph $G$ can be $O(\log n \log \log n)$-approximated by a probability distribution $\mathcal{D}_G$ of spanning trees of $G$. Furthermore, the distribution $\mathcal{D}_G$ is realizable by a probabilistic polynomial-time algorithm $\mathcal{A}$ (i.e., for every spanning tree $T$, $\Pr[\mathcal{A}(G) = T] = \mathbb{P}_{\mathcal{D}_G}[T]$).
Proposition 4.3 [7, 14]. Every Euclidean graph \( G \) embedded in \( \mathbb{R}^d \) can be \( O(d \log n) \)-approximated by a probability distribution \( \mathcal{D}_G \) of spanning trees of \( G \). Furthermore, the distribution \( \mathcal{D}_G \) is realizable by a probabilistic polynomial-time algorithm \( \mathcal{A} \).

Now, given a probability distribution \( \mathcal{D}_G \) of spanning trees of \( G \), one may randomly pick a tree according to \( \mathcal{D}_G \) and use it as the communication backbone of the arrow distributed directory protocol.

Theorem 4.4. If a graph \( G \) is \( \alpha \)-probabilistically approximated by a probability distribution \( \mathcal{D}_G \) of spanning trees, then the expected overhead ratio of the arrow protocol over a spanning tree \( T \) drawn randomly according to \( \mathcal{D}_G \) satisfies \( \bar{OH} \leq (\alpha + 1)/2 \).

Proof. The expected overhead ratio of the arrow protocol over a sequence of requests \( S = \langle v_{k_1}, \ldots, v_{k_l} \rangle \) is

\[
\bar{OH} = \mathbb{E} \left[ \frac{s(S, \mathcal{A})}{c(S, \mathcal{OPT})} \right] = \left( 1 + \mathbb{E} \left[ \frac{c(S, T)}{c(S, G)} \right] \right)/2.
\]

However, \( c(S, G) \) is independent of the selection of the tree \( T \), and since \( T \) is drawn randomly according to \( \mathcal{D}_G \),

\[
\mathbb{E}[c(S, T)] = \mathbb{E} \left[ \sum_{1 \leq i \leq l} d_T(v_i, v_{i+1}) \right] = \sum_{1 \leq i \leq l} \mathbb{E}[d_T(v_i, v_{i+1})] \\
\leq \sum_{1 \leq i \leq l} \alpha \cdot d_G(v_i, v_{i+1}) \leq \alpha \cdot c(S, G).
\]

Hence,

\[
\mathbb{E} \left[ \frac{s(S, \mathcal{A})}{c(S, \mathcal{OPT})} \right] \leq (1 + \alpha)/2.
\]

Corollary 4.5. Using a spanning tree drawn randomly as in Proposition 4.2, the expected overhead ratio of the arrow distributed directory protocol satisfies \( \bar{OH} = O(\log n \log \log n) \).

Corollary 4.6. In constant dimension Euclidean networks, using a spanning tree drawn randomly as in Proposition 4.3, the expected overhead ratio of the arrow distributed directory protocol satisfies \( \bar{OH} = O(\log n) \).
REFERENCES