


Towards Communication-Efficient Peer-To-Peer Networks

Khalid Hourani ✉ 

Department of Computer Science, University of Houston, TX, USA

William K. Moses Jr. ✉ 

Department of Computer Science, Durham University, UK

Gopal Pandurangan ✉ 

Department of Computer Science, University of Houston, TX, USA

Abstract

We focus on designing Peer-to-Peer (P2P) networks that enable efficient communication. Over the last two decades, there has been substantial algorithmic research on distributed protocols for building P2P networks with various desirable properties such as high expansion, low diameter, and robustness to a large number of deletions. A key underlying theme in all of these works is to distributively build a *random graph* topology that guarantees the above properties. Moreover, the random connectivity topology is widely deployed in many P2P systems today, including those that implement blockchains and cryptocurrencies. However, a major drawback of using a random graph topology for a P2P network is that the random topology does not respect the *underlying* (Internet) communication topology. This creates a large *propagation delay*, which is a major communication bottleneck in modern P2P networks.

In this paper, we work towards designing P2P networks that are communication-efficient (having small propagation delay) with provable guarantees. Our main contribution is an efficient, decentralized protocol, CLOSE-WEAVER, that transforms a random graph topology embedded in an underlying Euclidean space into a topology that also respects the underlying metric. We then present efficient point-to-point routing and broadcast protocols that achieve essentially optimal performance with respect to the underlying space.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Mathematics of computing → Probabilistic algorithms; Mathematics of computing → Discrete mathematics

Keywords and phrases Peer-to-Peer Networks, Overlay Construction Protocol, Expanders, Broadcast, Geometric Routing

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.71

Related Version *Full Version:* <https://arxiv.org/abs/2406.16661> [20]

Funding *Khalid Hourani:* K. Hourani was supported in part by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075 and BSF grant 2016419.

William K. Moses Jr.: Part of the work was done while William K. Moses Jr. was a postdoctoral fellow at the University of Houston, Houston, TX, USA. W. K. Moses Jr. was supported in part by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075 and BSF grant 2016419.

Gopal Pandurangan: G. Pandurangan was supported in part by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075 and BSF grant 2016419.

1 Introduction

There has been a long line of algorithmic research on building Peer-to-Peer (P2P) networks (also called overlay networks) with desirable properties such as connectivity, low diameter, high expansion, and robustness to adversarial deletions [35, 28, 16, 8, 9, 22, 5]. A key underlying theme in these works is a distributed protocol to build a *random graph* that guarantees these desirable properties. The high-level idea is for a node to connect to a



© Khalid Hourani, William K. Moses Jr., and Gopal Pandurangan;
licensed under Creative Commons License CC-BY 4.0

32nd Annual European Symposium on Algorithms (ESA 2024).

Editors: Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman; Article No. 71; pp. 71:1–71:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

small, but *random*, subset of nodes. In fact, this *random connectivity* mechanism is used in real-world P2P networks. For example, in the Bitcoin P2P network, each node connects to 8 neighbors chosen in a random fashion [30]. It is well-known that a random (bounded-degree) graph is an expander with high probability.¹² An expander graph on n nodes has high expansion and conductance, low diameter (logarithmic in the network size) and robustness to adversarial deletions – even deleting ϵn nodes (for a sufficiently small constant ϵ) leaves a giant component of $\Theta(n)$ size which is also an expander [19, 6].

Unfortunately, a major drawback of using a random graph as a P2P network is that the connections are made to random nodes and do not respect the *underlying* (Internet) communication topology. This causes a large *propagation latency* or *delay*. Indeed, this is a crucial problem in the Bitcoin P2P network, which has delays as high as 79 seconds on average [10, 30]. A main cause for the delay is that the P2P (overlay) network induced by random connectivity can be highly sub-optimal, since it ignores the underlying Internet communication topology (which depends on geographical distance, among other factors).³ The main problem we address in this paper is to show how one can efficiently modify a given random graph topology to build P2P networks that also have small *propagation latency*, in addition to other properties such as low (hop) diameter and high expansion, with provable guarantees.

1.1 Motivation, Model, and Definitions

Motivation. We consider a random graph network that is used in several prior P2P network construction protocols (e.g. [35, 28, 8, 5]). As mentioned earlier, real-world P2P networks, such as Bitcoin, also seek to achieve a random graph topology (which are expanders with high probability [32, 19]). Indeed, random graphs have been used extensively to model P2P networks (see e.g. [28, 35, 16, 8, 29]).

Before we formally state the model that is based on prior works [30, 11, 21], we explain the motivation behind it; we refer to [30] for more details and give a brief discussion here. Many of today’s P2P (overlay) networks employ the random connectivity algorithm; in fact, this is widely deployed in many cryptocurrency systems.⁴ In this algorithm, nodes maintain a small number of connections to other nodes chosen in a random fashion. In such a topology, for any two nodes u and v , any path (including the shortest path) would likely go through nodes that are not located close to the shortest geographical route (i.e., the geodesic) connecting u and v . Such paths that do not respect the underlying geographical placement of nodes often lead to higher propagation delay. Indeed, it can be shown that a random topology yields paths with propagation delays much higher than those of paths on topologies that respect the underlying geography [30].

To model the underlying propagation costs, several prior works (see e.g., the Vivaldi system [11]) have empirically shown that nodes on the Internet can be embedded on a low-dimensional metric space (e.g., \mathbb{R}^5) such that the distance between any two nodes accurately

¹ In this paper, by expander, we mean one with bounded degree, i.e., the degree of all nodes is bounded by a constant or a slow-growing function of n , say $O(\text{polylog } n)$, where n is the network size.

² For a graph with n nodes, we say that it has a property with high probability when the probability is at least $1 - 1/n^c$ for some $c \geq 1$.

³ It also ignores differences in bandwidth, hash-strength, and computational power across peers as well as malicious peers. Addressing these issues is beyond the scope of this paper.

⁴ In particular, the real-world Bitcoin P2P network, constructed by allowing each node to choose 8 random (outgoing) connections ([31, 30]) is likely an expander network if the connections are chosen (reasonably) uniformly at random [34].

captures the communication delay between them. In fact, the Vivaldi system demonstrates that even embedding the nodes in a 2-dimensional metric space (e.g., \mathbb{R}^2) and using the corresponding distances captures the communication delay quite well. In contrast, the paths on a random graph topology are highly sub-optimal, since they are unlikely to follow the optimal path on the embedded metric space.

The work of Mao et al. [30] illustrates the above disparity using the following example motivated by the above discussion. Consider a network embedded in the unit square $[0, 1] \times [0, 1]$. The set of nodes (points) V is drawn uniformly at random within the square. The Euclidean distance, $\|u - v\|_2$, between any two nodes $u, v \in V$ represents the *delay* or *latency* of sending a message from u to v (or vice-versa). We construct a *random graph* on V by connecting each node in the unit-square randomly to a small constant number of other nodes. Since the random graph does not respect the underlying geometry, the propagation cost between u and v – defined as the sum of the Euclidean distances of the edges in the shortest path – is significantly greater than the point-to-point (geodesic) distance ($\|u - v\|_2$) between them. We formally show this in Theorem 3.

By comparison, consider a *random geometric graph* on V , where the uniformly distributed nodes are connected as follows: any two nodes u and v are connected by an edge if they are within a distance ρ of each other [38, 33]. This model, called the $G(n, \rho)$ *random geometric graph* model, has connections which respect the underlying geometry. In this graph, we can show that the shortest path between any two nodes u and v is much closer to the geodesic shortest path (the straight line path) between u and v [33, 13].

We note that, while the work of Mao et al. [30] showcases the disparity between a random graph and a random geometric graph (as discussed above) it does not give any theoretical results on how to convert a random graph topology into a random geometric graph topology. On the other hand, it gives heuristics to transform a P2P graph constructed on real-world data to a graph that has smaller propagation delays. The heuristics are based on rewiring edges to favor edges between nodes that have smaller round-trip delays. It presents experimental simulations to show that these heuristics do well in practice. However, they do not formally analyze their algorithm and do not give any theoretical guarantees.

Network Model. Motivated by the above discussion, and following prior works [30, 11, 21], we model a P2P network G as follows. We assume G to be a d -regular expander (where d is a constant).⁵ Note that our results will also apply if G has a random connectivity topology modeled by a d -regular (or bounded degree) *random graph* or a $G(n, p)$ random graph (with $p = \Theta(\log n/n)$). We note that such random graphs are expanders with high probability (see Definition 9) [26]. Our model is quite general in the sense that we only assume that the topology is an expander; no other special properties are assumed. Furthermore, we assume that the nodes of G correspond to points that are distributed *uniformly at random* in a unit square $[0, 1] \times [0, 1]$.⁶ Although, the assumption of nodes being uniformly distributed is strong, based on our experiments on the Bitcoin P2P network, this appears to be a reasonable first approximation.⁷ Considering more general distribution models is a good direction for future work (cf. Section 1.4).

⁵ We assume a d -regular graph for convenience; we could have also assumed that the degree is bounded by some small growing function of n , say $O(\log n)$.

⁶ Our model can be generalized to higher dimensions by embedding nodes in an m -dimensional hypercube $[0, 1]^m$.

⁷ We embedded nodes in the Bitcoin P2P network in a 2-dimensional grid using the Vivaldi algorithm and although there were many outliers, a significant subset of nodes ended up being reasonably uniformly distributed.

We assume each node u knows its ID and, while node u *need not* know its coordinates, it is able to determine its distance (which captures propagation delay) to any node v given only the ID of v .⁸ In particular, we assume for convenience that a node can determine the Euclidean and the Manhattan distances (i.e., L_2 and L_∞ norms respectively) between itself and another node if it knows the ID of that node.

An important assumption is that nodes initially have only *local* knowledge, i.e., they have knowledge of only themselves and their neighbors in G . In particular, they do *not* have any knowledge of the global topology or of the IDs of other nodes (except those of their neighbors) in the network. We assume that nodes have knowledge of the network size n (or a good estimate of it).

We assume a synchronous network where computation and communication proceeds in a sequence of discrete *rounds*. Communication is via message passing on the edges of G . Note that G is a P2P (overlay) network in the sense that a node u can communicate (directly) with another node v if u knows the ID of v . This is a typical assumption in the context of P2P and overlay networks, where a node can establish communication with another node if it knows the other node's IP address, and has been used in several prior works (see e.g. [5, 4, 22, 36, 17]). Note that u can know the ID of v either directly, because u and v are neighbours in G , or indirectly, through received messages. In the latter case, this is equivalent to adding a “*virtual*” edge between u and v . Since we desire efficient protocols, we require each node to send and receive messages of size at most $\text{polylog}(n)$ bits in a round. In fact, a node will also communicate with only $\text{polylog}(n)$ other nodes in a round. Additionally, the number of bits sent per edge per round is $O(\text{polylog}(n))$.

1.2 Preliminaries

We need the following concepts before we formally state the problem that we address and our contributions.

Embedded Graph. We define an *embedded graph* as follows.

► **Definition 1.** Let $G = (V, E)$ be any graph and consider a random embedding of the nodes V into the unit square, i.e., a uniform and independent assignment of coordinates in $[0, 1] \times [0, 1]$ to each node in V . This graph, together with this embedding, is called an *embedded graph*, and we induce weights on the edge set E , with the weight of an edge (u, v) equal to the Euclidean distance between the coordinates assigned to u and v , respectively.

Routing Cost. We next define *propagation cost* to capture the cost of routing along a path in an embedded graph.

► **Definition 2.** Let G be an embedded graph. For any path $P = (v_1, v_2, \dots, v_{k-1}, v_k)$ the *propagation cost*, also called the *routing propagation cost*, of the path P is the weight of the path P given by $d_G(P) = \sum_{i=1}^{k-1} d(v_i, v_{i+1})$, i.e., the sum of the weights (the Euclidean distances) of edges along the path. The value k , denoted by $\text{hopcount}_G(P)$, is the *hop count* (or *hop length*) of the path P . The minimum propagation cost between the nodes u and v is the weight of the shortest path between u and v in the embedded graph.

⁸ Note that this assumption has to do only with implementing our protocols in a localized manner (which is relevant in practice) and does not affect their correctness or efficiency. In the Internet, for example, point-to-point propagation delay can be measured locally: a node can determine the round-trip-time to another node using the `ping` network utility [7]. On the other hand, it is also possible for a node to determine its coordinates – as mentioned earlier, systems such as Vivaldi [11] can assign coordinates in a low dimensional space (even \mathbb{R}^2) that accurately capture the propagation delay between nodes.

Note that the propagation cost between two nodes is lower bounded by the Euclidean distance between them. Given two nodes, we would like to route using a path of *small* propagation cost, i.e., a path whose propagation cost is close to the Euclidean distance between the two nodes. In particular, we would like the ratio between the two to be small. (We would also like the hop count to be small.)

The following theorem shows that, in a d -regular random graph G embedded in the unit square, the *ratio* of the propagation cost of the shortest path between two nodes u and v in G to the Euclidean distance between those nodes can be as high as $\Omega(\sqrt{n})$ on expectation. Thus a P2P topology that is modeled by a random graph topology has a high propagation cost for some node pairs.

► **Theorem 3.** *Let G be a d -regular random graph embedded in the unit square. Then, there exists a pair of nodes u and v in G such that $d_G(P)/d(u, v)$, the ratio of the propagation cost of the shortest path P between u and v to the Euclidean distance between them is $\Omega(\sqrt{n})$ on expectation.*

We use propagation cost to measure the performance of a routing algorithm in G . The goal is to construct a graph topology so that one can find paths of small propagation costs between every pair of nodes. Moreover, we want a routing algorithm that routes along paths of small propagation cost while also keeping the hop length small.

Broadcast Performance Measures. Next, we quantify the performance of a broadcasting algorithm.

► **Definition 4.** *Consider a broadcast algorithm \mathcal{A} that broadcasts a single message from a given source to all other nodes in some connected embedded graph G . The broadcast propagation cost of algorithm \mathcal{A} on graph G is defined as the the sum of the Euclidean distance of the edges used by \mathcal{A} to broadcast the message.*

Notice that the broadcast propagation cost roughly captures the efficiency of a broadcast algorithm. We note that the best possible broadcast propagation cost for a graph is broadcasting by using *only* the edges of the minimum spanning tree (MST) on G . In particular, this yields the following lower bound for a graph whose nodes are embedded uniformly at random in the unit square. The proof follows from a bound on the weight of a Euclidean MST on a set of points distributed uniformly in a unit square [2].

► **Theorem 5** (follows from [2]). *The broadcast propagation cost of any algorithm \mathcal{A} on an embedded graph G whose nodes are distributed uniformly in a unit square is $\Omega(\sqrt{n})$ with high probability.*

On the other hand, we show that the broadcast propagation cost of the standard flooding algorithm [37] on a random graph embedded in a unit square is high compared to the above lower bound.

► **Theorem 6.** *Let G be a d -regular random graph embedded in the unit square. The standard flooding algorithm on G has $\Theta(n)$ expected broadcast propagation cost.*

We also use other metrics to measure the quality of a broadcast algorithm \mathcal{A} . The *broadcast completion cost* and *broadcast completion time* measure, respectively, the propagation cost and the number of hops needed to reach any other node v from a given source u .

► **Definition 7.** Consider a broadcast algorithm \mathcal{A} that broadcasts a single message from some source node z to all other nodes in some connected graph $G(V, E)$. The broadcast completion cost of \mathcal{A} on G is the maximum value of the minimum propagation cost between the source node s and any node u considering paths taken by the message in \mathcal{A} , taken over all nodes $u \in V$ and all possible source nodes $s \in V$. More precisely, let $\text{Prop}_{\mathcal{A}}(s, u)$ be the minimum propagation cost for a message sent from the node s to reach node u using broadcast algorithm \mathcal{A} and define $\text{Prop}_{\mathcal{A}}(s) = \max_{u \in V} \text{Prop}_{\mathcal{A}}(s, u)$. Then, broadcast completion cost is $\max_{s \in V} \text{Prop}_{\mathcal{A}}(s)$. The broadcast completion time of \mathcal{A} on G is simply the number of rounds before the message from the source node reaches all nodes.

Conductance and Expanders. We recall the notions of conductance of a graph and that of an expander graph.

► **Definition 8 (Conductance).** The conductance $\phi(G)$ of a graph $G = (V, E)$ is defined as: $\phi(G) = \min_{S \subseteq V} \frac{|E(S, \bar{S})|}{\min\{\text{Vol } S, \text{Vol } \bar{S}\}}$ where, for any set S , $E(S, \bar{S})$ denotes the set of all edges with one vertex in S and one vertex in $\bar{S} = V - S$, and $\text{Vol}(S)$, called the volume of S , is the sum of the degrees of all nodes in S .

► **Definition 9 (Expander Graph).** A family of graphs G_n on n nodes is an expander family if, for some constant α with $0 < \alpha < 1$, the conductance $\phi_n = \phi(G_n)$ satisfies $\phi_n \geq \alpha$ for all $n \geq n_0$ for some $n_0 \in \mathbb{N}$.

Random Geometric Graph.

► **Definition 10 (Random Geometric Graph).** A random geometric graph, $G(n, \rho) = (V, E)$, is a graph of n points, independently and uniformly at random placed within $[0, 1] \times [0, 1]$ (the unit square). These points form the node set V , and for two nodes u and v , $(u, v) \in E$ if and only if the distance $d(u, v)$ is at most ρ , for parameter $0 < \rho = f(n) \leq 1$.

We note that the distance between points is the standard Euclidean distance. The $G(n, \rho)$ graph exhibits the threshold phenomenon for many properties, such as connectivity, coverage, presence of a giant component, etc. [38, 33]. For example, the threshold for connectivity is $\rho = \Theta(\sqrt{\log n/n})$, i.e., if the value of ρ is $\Omega(\sqrt{\log n/n})$, the graph $G(n, \rho)$ is connected with high probability; on the other hand, if $\rho = o(\sqrt{\log n/n})$, then the graph is likely to be disconnected. It is also known [14] that the diameter of $G(n, \rho)$ (above the connectivity threshold) is $\tilde{O}(1/\rho)$ with high probability.⁹

1.3 Problems Addressed and Our Contributions

As shown in Theorems 3 and 6, routing (even via the shortest path) and the standard flooding broadcast protocol in an embedded *random* graph G have a relatively large point-to-point routing propagation cost and broadcast propagation cost, respectively.

Given a P2P network modeled as a random graph G embedded on a unit square, the goal is to design an efficient distributed protocol to transform G into a network G^* that admits efficient communication primitives for the fundamental tasks of routing and broadcast, in particular, those that have essentially optimal routing and broadcast propagation costs. Furthermore, we want to design optimal routing and broadcast protocols on G^* . (Broadcasting is a key application used in P2P networks that implement blockchain and cryptocurrencies in which a block must be quickly broadcast to all (or most) nodes in the network.)

⁹ Throughout, the \tilde{O} notation hides a polylog n factor and $\tilde{\Omega}$ hides a $1/(\text{polylog } n)$ factor.

Our contributions are as follows:

1. We develop a theoretical framework to model and analyze P2P network protocols, specifically point-to-point routing and broadcast (see Section 1.1).
2. We present an efficient distributed P2P topology construction protocol, CLOSE-WEAVER, that takes a P2P expander network G and improves it into a topology G^* that admits essentially optimal routing and broadcast primitives (see Section 2). Our protocol uses only local knowledge and is fast, using only $O(\text{polylog } n)$ rounds. CLOSE-WEAVER is based on random walks which makes it quite lightweight (small local computation overhead) and inherently decentralized and robust (no single point of action, no construction of tree structure, etc). It is also scalable in the sense that each node sends and receives only $O(\text{polylog } n)$ bits per round and communicates with only $O(\text{polylog } n)$ nodes at any round. We assume only that the given topology G is an expander graph; in particular, G can be random graph (modeling a random connectivity topology, see Section 1.1).
3. To show the efficiency of G^* , we develop a distributed routing protocol GREEDY-ROUTING as well as broadcast protocols GEOMETRIC-FLOODING and COMPASS-CAST that have essentially optimal routing and broadcast propagation costs, respectively (see Section 3).

For lack of space, we refer to the full paper for proofs, additional details, and figures.

1.4 Other Related Work

There are several works (see e.g., [3, 15, 17]) that begin with an arbitrary graph and reconfigure it to be an expander (among other topologies). The expander topology constructed does not deal with the underlying (distance) metric. Our work, on the other hand, starts with an arbitrary expander topology (and here one can use algorithms such as the one in these papers to construct an expander overlay to begin with) and reconfigures it into an expander that also optimizes the propagation delay with respect to the underlying geometry. Thus our work can be considered as orthogonal to the above works.

There has been significant amount of work on a related problem, namely, constructing distributed hash tables (DHTs) and associated search protocols that respect the underlying metric [40, 39, 23, 1, 12, 18]. In this line of work, nodes store data items and they can also search for these items. The cost of the search, i.e., the path a request takes from the requesting node to the destination node, is measured with respect to an underlying metric. The goal is to build an overlay network and a search algorithm such that the cost of all paths is close to the metric distance. Our work is broadly in same spirit as these works, with a key difference. While the previous works build an overlay network while assuming global knowledge of costs between all pairs of nodes, our work assumes that we start with a sparse (expander) topology with only local knowledge of costs (between neighbors only), which is more realistic in a P2P network. Furthermore, in these works, the underlying metric is assumed to be *growth-restricted* which is more general than the 2-dimensional plane assumed here. In a growth-restricted metric, the ball of radius $2r$ around a point x contains at most a (fixed) constant fraction of points more than the ball of radius r around x . This is more general than the uniform distribution in a 2-dimensional plane assumed here (which is a special case of growth-restricted) since, in a growth-restricted metric, points need not be uniformly dense everywhere. An interesting direction of future work is extending our protocols to work in general growth-restricted metrics.

Routing protocols (that are similar in spirit to ours) that assume that each node knows its position and that of its neighbors and that the position of the destination is known to the source are sometimes referred to as geometric routing and greedy approaches to such routing have been explored extensively in the literature (e.g. [25], [24], [27] and the references therein).

2 Close-Weaver: A P2P Topology Construction Protocol

We show how to convert a given d -regular (d is a constant) expander graph embedded in the Euclidean plane (Definition 1) into a graph that, in addition to having the desired properties of an expander, also allows more efficient routing and broadcasting with essentially optimal propagation cost.¹⁰ The main result of this section is the CLOSE-WEAVER protocol, running in polylog n rounds, that yields a network with $O(\log^2 n)$ degree and contains a *random geometric graph* as a subgraph.

2.1 The Protocol

Brief Description. Starting with an embedded, d -regular expander $G = (V, E)$, the algorithm constructs a series of expander graphs, one per phase, such that in each phase i , each node u connects to some $O(\log n)$ random neighbors located in a square (box) of side-length r^i centered at u (that intersects the unit square), where $0 < r < 1$ is a fixed constant (we can fix $r = 1/4$ due to technical considerations in Section 3.1). In the final phase, κ , each node u connects to all $O(\log n)$ neighbors contained in the square of side length r^κ at its center. In this manner, we construct a final graph, which is the union of the original graph and all graphs constructed in each phase, which has low degree ($O(\log^2 n)$) and low diameter ($O(\log n)$). We note that we require $r^{2\kappa}n = \Theta(\log n)$, and hence $\kappa = \Theta(\log n - \log \log n)/(\log 1/r)$.

Our protocol makes extensive use of random walks and the following lemma is useful in bounding the rounds needed to perform many random walks in parallel under the bandwidth constraints (polylog n bits per edge per round).

► **Lemma 11** (Adapted from Lemma 3.2 in [42]). *Let $G = (V, E)$ be an undirected graph and let each node $v \in V$, with degree $\deg(v)$, initiate $\eta \deg(v)$ random walks, each of length λ . Then all walks finish their respective λ steps in $O(\eta\lambda \log n)$ rounds with high probability.*

Detailed Description. Let $B_u(\ell)$ denote the *intersection* of the unit square (recall that the Euclidean plane is constrained to a square grid of side length 1) and the square of side-length ℓ centered at node u . Note that if u is located at least distance $\ell/2$ from every edge of the grid, then $B_u(\ell)$ is merely the square with side-length ℓ centered on u . Run the following algorithm for $\kappa = c \log n$ phases, for appropriately chosen constant c , starting from phase 1. The first $\kappa - 1$ phases are described below and the final phase is described subsequently.

In each phase $1 \leq i \leq \kappa - 1$, we associate a graph with each node u that contains all nodes and their associated edges inside $B_u(r^i)$ created in phase i – which we denote by $G_u(i)$. Denote the initial graph for a node u by $G_u(0)$ (note that $G_u(0) \equiv G$). Define $G(i) = \cup_{u \in V} G_u(i)$, i.e., the union of these graphs across all nodes (note that $G(0) \equiv G$).

¹⁰As mentioned earlier, our protocol will also work for d -regular random graphs which are expanders with high probability. Also, the graph need not be regular; it is enough if the degree is bounded, say $O(\text{polylog } n)$, to get the desired performance bounds.

First $\kappa - 1$ phases: Each phase $i \in \{1, 2, \dots, \kappa - 1\}$, consists of two major steps outlined below: (Note that we assume at the beginning of phase i , graphs $G_u(i - 1)$ have been constructed for all u .) In phase i , we construct $G_u(i)$ for all u using lazy random walks.

- (1) For each node u , perform $\Theta(\log n)$ *lazy random walks* of length 2τ , where $\tau = a \log n$ (for a constant a sufficiently large to guarantee rapid mixing, i.e., reaching close to the stationary distribution), in $G_u(i - 1)$, which is assumed to be an expander (this invariant will be maintained for all i).

A lazy random walk is similar to a normal random walk except that, in each step, the walk stays at the current node u with probability $1 - \deg(u)/(\Delta + 1)$, otherwise it travels to a random neighbor of u (in $G_u(i - 1)$, i.e., in box $B_u(r^{i-1})$). Here, $\deg(u)$ is the degree of u and Δ is an upper bound on the maximum degree, which is $O(\log n)$ in $G_u(i - 1)$ (by protocol design). We maintain the ratio $\deg(u)/(\Delta + 1) = O(1)$ in every phase (by protocol design), hence the slowdown of the lazy random walk (compared to the normal random walk) is at most a constant factor. It is known that the stationary distribution of a lazy random walk is *uniform* and such a walk, beginning at a node u , will arrive at a fixed node v in $G_u(i - 1)$ with probability $1/n \pm 1/n^3$ after τ number of steps [41]. Thus, a lazy random walk from u gives a way to sample a node *nearly uniformly* at random from the graph $G_u(i - 1)$. Each lazy random walk starting from u is represented by a token containing the ID of u , the current phase number, and the number of steps remaining in the lazy random walk; in phase i , this token is passed from node to node to simulate a random walk within $B_u(r^{i-1})$.¹¹

Note that each node v that receives the token only considers the subset of its neighbors that are within $B_u(r^{i-1})$ when considering nodes to pass the token to. After 2τ steps, if the token lands within $B_u(r^i)$, the random walk is *successful*. By Lemma 11, all walks will finish in $O(\log^3 n)$ rounds in the first phase and $O(\log^2 n)$ rounds in subsequent phases with high probability. Note that, to maintain synchronicity, all nodes participate and wait for $O(\log^3 n)$ rounds to finish in the first phase and $O(\log^2 n)$ rounds in subsequent phases.

- (2) The graph $G_u(i) = (V_u(i), E_u(i))$ is constructed as follows: its node set $V_u(i)$ is the set of all nodes in the box $B_u(r^i)$. Edges from nodes in $V_u(i)$ are determined as follows. Suppose a lazy random walk from a node $x \in V_u(i)$ successfully ends at y , i.e., y is within the box of $B_x(r^i)$ (note that, unless $u = x$, this box is different from $B_u(r^i)$, but does overlap with at least $1/4$ of $B_u(r^i)$). Node y will send a message to x informing it that its random walk successfully terminated at y . Among all such nodes that notify x , x will sample (without replacement) a subset of $b \log n$ nodes (for a fixed constant b) and add undirected edges to these sampled nodes. The edge set $E_u(i)$ of $G_u(i)$ consists only of edges between nodes in $V_u(i)$.

Last phase: The final phase is similar, except that each node u initiates a larger number of random walks, so that with high probability *all nodes* within the box $B_u(r^\kappa)$ (note that $r^\kappa = \Theta(\sqrt{\log n/n})$) are sampled and thus u is able to form connections to all nodes in $B_u(r^\kappa)$ (which contains $\Theta(\log n)$ nodes). This will ensure that $G(\kappa) = \cup_{u \in V} G_u(\kappa)$ contains a random geometric graph $G(n, \rho)$ with $\rho = \Theta(\sqrt{\log n/n})$. More precisely, in the final phase (phase κ), each node u runs $\Theta(\log^2 n)$ random walks on $G(\kappa - 1)$ to all nodes within $B_u(r^\kappa)$ to form graph $G(\kappa)$.

¹¹As assumed in Section 1.1, a node on the random walk path can check whether it is within the box $B_u(r^{i-1})$ centered at the source node u , since it knows the ID of u and hence the L_∞ distance from u .

The final graph G^* is the union of the graphs $G(i)$, $0 \leq i \leq \kappa$. Algorithm 1 gives a high-level summary of the protocol.

■ **Algorithm 1** CLOSE-WEAVER Construction Protocol.

```

1: for each node  $u$  and phase  $i$  in  $\{1, 2, \dots, \kappa - 1\}$  do
2:    $u$  performs  $\Theta(\log n)$  random walks of length  $2\tau = \Theta(\log n)$  in  $G_u(i - 1)$ 
3:    $u$  connects to  $\Theta(\log n)$  nodes where random walks are successful
4: end for each
5: for each node  $u$  do
6:    $u$  initiates  $\Theta(\log^2 n)$  lazy random walks in  $B_u(r^\kappa)$  and connects to nodes where walk ends
   successfully
7: end for each

```

2.2 Protocol Analysis

We prove that, with high probability, the protocol takes $O(\log^3 n)$ rounds and constructs a graph G^* that has maximum degree $O(\log^2 n)$ and contains a random geometric graph as a subgraph (besides being an expander).

To argue that the constructed graph G^* contains a random geometric graph, we show that the series of graph constructions proceeds correctly in each phase, resulting in the last phase constructing $G(\kappa)$, the desired random geometric graph. Each phase i crucially relies on the fact that the subgraph induced by a given node u in phase $i - 1$, $G_u(i - 1)$, is an expander. In each phase i , we perform several lazy random walks starting from each node u on $G_u(i - 1)$. Since the lengths of lazy random walks starting at u performed on $G_u(i - 1)$ are $\Omega(\log n)$, we see that they run longer than the mixing time of $G_u(i - 1)$, resulting in the final destination of the walk, i.e., the neighbor of u in that phase resulting from the random walk, being chosen uniformly at random from the vertices of $G_u(i - 1)$. This property is useful in the analysis. The random walks performed by each node u in phase i result in at least $\Omega(\log n)$ neighbors that can be used by u to construct its part of the graph $G(i)$. Finally, with high probability, the subgraph induced by edges of length less than r^κ forms a random geometric graph (see full version).

To argue about the maximum degree of the graph, notice first that we construct at most $\kappa = O(\log n)$ subgraphs, one per phase. By showing that each of these subgraphs has a maximum degree of $O(\log n)$ with high probability, we show that the maximum degree of the graph is $O(\log^2 n)$ with high probability. The degree of any node in $G(i)$ does not exceed $O(\log n)$, for all phases i excluding the final phase κ (see full paper). In our analysis, we bound the number of nodes in a box surrounding a given node, and in particular show that the degree of each node in the final phase does not exceed $O(\log n)$ with high probability, i.e., the maximum degree of the graph $G(\kappa)$ is $O(\log n)$.

All these properties of the final graph G^* are captured by Theorem 12. We argue about the run time directly in the proof of Theorem 12.

The key lemma is showing that each graph $G(i)$ formed at the end of each phase i is an expander. It can be proved by induction on i . The base case is given, since $G_u(0) \equiv G$ and G is an expander. For the induction hypothesis, we assume that $G_u(i - 1)$ is an expander and prove that $G_u(i)$ is an expander as well. The main technical idea behind the proof is to show that, with high probability, every subset of nodes (that is of size at most half the size of $V_u(i)$) has a conductance that is at least some fixed constant. The protocol initiates random-walks by each node in each phase of the algorithm to construct an expander, and the random walks occur over different subgraphs (regions). This makes it non-trivial to show that the constructed subgraph around each node is an expander at each phase. Since each

node does random walks in a local region around itself, the expansion proof has to be done carefully. To save space, we leave the required lemmas and proof of Theorem 12 for the full version of the paper.

► **Theorem 12.** *The CLOSE-WEAVER protocol (Algorithm 1) takes an embedded d -regular expander graph and constructs a graph in $O(\log^3 n)$ rounds such that:*

- (i) *its degree is $O(\log^2 n)$ with high probability*
- (ii) *and it contains a random geometric graph $G(n, \rho)$ (where $\rho = \Theta(\sqrt{(\log n)/n})$) with high probability.*

3 Efficient Communication Protocols

In this section, we present efficient routing and broadcast algorithms for the graph G^* that was constructed using the P2P protocol CLOSE-WEAVER in Section 2. Since the properties of G^* hold with high probability, the correctness of the protocols and the associated bounds in the theorems hold with high probability.

3.1 Efficient Broadcasting Protocols

Let us assume that we are given a source node *source* with a message that must be broadcast to every node in the graph. In this section, we design broadcast algorithms to be run on the graph G^* that is constructed by the P2P construction protocol in Section 2. In order to argue about the efficiency of broadcast, we use *broadcast propagation cost*, *broadcast completion cost*, and *broadcast completion time* (see Section 1.2).

First, we present a simple flooding-based broadcast algorithm called GEOMETRIC-FLOODING, in Section 3.1.1, that has optimal broadcast propagation cost (up to polylog n factors) and optimal broadcast completion cost (up to polylog n factors) but at the expense of a very bad broadcast completion time. In particular, the broadcast propagation cost is $\tilde{O}(\sqrt{n})$, the broadcast completion cost is $\tilde{O}(1)$, and the broadcast completion time is $\tilde{O}(\sqrt{n})$. From Theorem 5, we see that this broadcast propagation cost is asymptotically optimal up to polylog n factors for any broadcast algorithm run by nodes uniformly distributed in Euclidean space.

In order to obtain optimal bounds (up to polylog n factors) for all three metrics, we design a more sophisticated algorithm called COMPASS-CAST, in Section 3.1.2, that requires that each node knows its own coordinates (instead of merely the distance between itself and some other node). COMPASS-CAST has broadcast propagation cost $\tilde{O}(\sqrt{n})$, broadcast completion cost $O(1)$, and broadcast completion time $\tilde{O}(1)$.

3.1.1 Algorithm Geometric-Flooding

Brief Description. The algorithm consists of each node participating in flooding over $G(\kappa)$. Initially, the source node sends the message to all its neighbors in $G(\kappa)$. Subsequently, each node, once it receives the message for the first time, transmits that message over each of its edges in $G(\kappa)$.

Analysis. In $G(\kappa)$, each node has $O(\log n)$ neighbors and the weight of each edge is $O(r^\kappa)$. So, the sum of the edge weights in the graph, i.e., the broadcast propagation cost, is $O(n \cdot \log n \cdot r^\kappa) = O(n \cdot \log n \cdot \sqrt{\log n/n}) = O(\sqrt{n} \log^3 n)$.

The broadcast completion time corresponds to the diameter of the random geometric graph $G(\kappa)$. From [14], we see that the diameter of a random geometric graph $G(n, \rho)$ embedded in a unit grid is $\tilde{\Theta}(1/\rho)$. For the graph $G(\kappa)$, $\rho = \Theta(\sqrt{\log n/n})$. So the broadcast completion time is $\tilde{\Theta}(\sqrt{n})$.

The broadcast completion cost is upper bounded by the product of diameter and edge weight, so it is $\tilde{O}(1)$.

The following theorem captures the relevant properties of the algorithm.

► **Theorem 13.** *Algorithm GEOMETRIC-FLOODING, when run by all nodes on G^* , results in a message being sent from a source node source to all nodes in $\tilde{O}(\sqrt{n})$ broadcast completion time with broadcast completion cost $\tilde{O}(1)$ and broadcast propagation cost $O(\sqrt{n \log^3 n})$, which are all asymptotically optimal up to polylog n factors.*

3.1.2 Algorithm Compass-Cast

Note that for this section, due to technical considerations, we assume that the parameter r in the P2P construction protocol is chosen so that $r \leq 0.25$ and $1/r$ is an integer. We additionally assume that nodes know their own coordinates.

In order to describe the algorithm, we make use of the following notation for ease of explanation. Let H_i represent the partition of the unit grid into a $1/r^i$ by $1/r^i$ grid of $1/r^{2i}$ equal size squares.

Brief Description. The efficient broadcast of a message can be done in three phases. In phase one, the message is propagated to exactly one node in each square of H_2 using $G(1)$. Phase two is used to propagate the message to exactly one node in each square in H_κ in a recursive manner as follows. Each node that received a message at the end of phase one takes “ownership” of all square of H_3 that lie within its square of H_2 and sends the message to exactly one node in each such square of H_3 . In this manner, each node u with the message in a square in H_i , $i \leq 2 < \kappa$, chooses one node per square of H_{i+1} that lies within u 's square of H_i and sends the message to them. Finally, exactly one node in each square of H_κ will have the message. Phase three is used to propagate the message to every node in G^* by having each node in the proceedings phase transmit the message to all its neighbors in G^* . Subsequently, each node that received the message further transmits it to all its neighbors in G^* .

To save space, the analysis of algorithm COMPASS-CAST, which yields Theorem 14, is left for the full version.

► **Theorem 14.** *Algorithm COMPASS-CAST, when run by all nodes on G^* , results in a message being sent from a source node source to all nodes with broadcast completion cost $O(1)$, and $O(\log n)$ broadcast completion time and broadcast propagation cost $O(\sqrt{n \log^3 n})$, which are asymptotically optimal up to polylog n factors.*

3.2 An Efficient Routing Protocol

In this section, we present an efficient routing algorithm, Algorithm GREEDY-ROUTING (pseudocode in Algorithm 2), which allows us to route a packet from any source S to any destination F in $O(\log n)$ hops using G^* such that the path taken has propagation cost $O(d(S, F))$, where $d(S, F)$ is the Euclidean distance between S and F . An important property of this routing protocol is that it is localized and greedy: any node needs only local information (of itself and its neighbors) to route a given message to its final destination.

Due to a lack of space, we give only the theorem statement below. A full analysis may be found in the full version.

► **Theorem 15.** *Consider the graph G^* obtained at the end of Algorithm 1. For any source node S and any destination node F , routing a packet from S to F using Algorithm 2 takes $O(\log n)$ hops and the propagation cost of the routed path is $O(d(S, F))$, where $d(S, F)$ is the Euclidean distance between S and F .*

■ **Algorithm 2** Greedy Routing – forwarding a message m from node S to node F .

```

1: current = S
2: dist = ∞
3: while current ≠ F do
4:   Send a message to every neighbor of current requesting  $d(\text{neighbor}, F)$ 
5:   for each neighbor of current do
6:     new-dist =  $d(\text{neighbor}, F)$ 
7:     if new-dist < dist then
8:       dist = new-dist
9:       closest-neighbor = neighbor
10:    end if
11:  end for each
12:  forward message to closest-neighbor (which then becomes current)
13: end while

```

4 Conclusion and Future Work

We consider this work as a theoretical step towards the design and analysis of P2P topologies and associated communication protocols. While our theoretical framework is only a rough approximation to real-world P2P networks, it provides a rigorous model for the design and analysis of P2P protocols that takes into account propagation delays that depend on not only the graph topology but also on the distribution of nodes across the Internet. Our model is inspired by several studies on the Internet, particularly the Vivaldi system [11], which posits how nodes on the Internet can be assigned coordinates in a low-dimensional, even 2-dimensional, Euclidean space, that quite accurately captures the point-to-point latencies between nodes. We have additionally performed empirical research, via simulation, on the Bitcoin P2P network that suggests that the model is a reasonable approximation to a real-world P2P network. We have also performed preliminary simulations of our routing and broadcast protocols which broadly support our theoretical bounds. We leave a detailed experimental study for future work.

An open problem would be to devise a protocol that has similar guarantees to CLOSE-WEAVER, but with a better degree guarantee, such as a constant degree.

References

- 1 Ittai Abraham, Dahlia Malkhi, and Oren Dobzinski. LAND: stretch $(1 + \epsilon)$ locality-aware networks for dhts. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 550–559. SIAM, 2004. URL: <http://dl.acm.org/citation.cfm?id=982792.982873>.
- 2 D. Aldous and J.M. Steele. Asymptotics for euclidean minimal spanning trees on random points. *Probab. Th. Rel. Fields*, (92):247–258, 1992.
- 3 Dana Angluin, James Aspnes, Jiang Chen, Yinghua Wu, and Yitong Yin. Fast construction of overlay networks. In Phillip B. Gibbons and Paul G. Spirakis, editors, *SPAA 2005: Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures, July 18-20, 2005, Las Vegas, Nevada, USA*, pages 145–154. ACM, 2005. doi:10.1145/1073970.1073991.
- 4 John Augustine, Gopal Pandurangan, and Peter Robinson. Fast byzantine agreement in dynamic networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 74–83, 2013.
- 5 John Augustine, Gopal Pandurangan, Peter Robinson, Scott Roche, and Eli Upfal. Enabling robust and efficient distributed computation in dynamic peer-to-peer networks. In *IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 350–369, 2015.

- 6 Amitabha Bagchi, Ankur Bhargava, Amitabh Chaudhary, David Eppstein, and Christian Scheideler. The effect of faults on network expansion. *Theory Comput. Syst.*, 39(6):903–928, 2006. doi:10.1007/s00224-006-1349-0.
- 7 Free BSD. ping(8), 2022. URL: <https://man.freebsd.org/cgi/man.cgi?query=ping&sektion=8>.
- 8 Colin Cooper, Martin E. Dyer, and Catherine S. Greenhill. Sampling regular graphs and a peer-to-peer network. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 980–988. SIAM, 2005.
- 9 Colin Cooper, Martin E. Dyer, and Catherine S. Greenhill. Sampling regular graphs and a peer-to-peer network. *Combinatorics, Probability & Computing*, 16(4):557–593, 2007.
- 10 Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. In Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 106–125, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
- 11 Frank Dabek, Russ Cox, M. Frans Kaashoek, and Robert Tappan Morris. Vivaldi: a decentralized network coordinate system. In Raj Yavatkar, Ellen W. Zegura, and Jennifer Rexford, editors, *ACM SIGCOMM*, pages 15–26, 2004.
- 12 Maximilian Drees, Robert Gmyr, and Christian Scheideler. Churn- and dos-resistant overlay networks based on network reconfiguration. In Christian Scheideler and Seth Gilbert, editors, *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 417–427. ACM, 2016. doi:10.1145/2935764.2935783.
- 13 Tobias Friedrich, Thomas Sauerwald, and Alexandre Stauffer. Diameter and broadcast time of random geometric graphs in arbitrary dimensions. *Algorithmica*, 67(1):65–88, 2013.
- 14 Ghurumuruhan Ganesan. Stretch and diameter in random geometric graphs. *Algorithmica*, 80:300–330, 2018.
- 15 Seth Gilbert, Gopal Pandurangan, Peter Robinson, and Amitabh Trehan. Dconstructor: Efficient and robust network construction with polylogarithmic overhead. In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, pages 438–447. ACM, 2020. doi:10.1145/3382734.3405716.
- 16 C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks: Algorithms and evaluation. *Performance Evaluation*, 63(3):241–263, 2006.
- 17 Thorsten Götte, Kristian Hinnenthal, Christian Scheideler, and Julian Werthmann. Time-optimal construction of overlay networks. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 457–468, 2021.
- 18 Kirsten Hildrum, John Kubiawicz, Sean Ma, and Satish Rao. A note on the nearest neighbor in growth-restricted metrics. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 560–561. SIAM, 2004. URL: <http://dl.acm.org/citation.cfm?id=982792.982874>.
- 19 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.
- 20 Khalid Hourani, William K. Moses Jr., and Gopal Pandurangan. Towards communication-efficient peer-to-peer networks, 2024. arXiv:2406.16661.
- 21 Zi Hu, John S. Heidemann, and Yuri Pradkin. Towards geolocation of millions of IP addresses. In *Proceedings of the 12th ACM SIGCOMM Internet Measurement Conference, IMC*, pages 123–130, 2012.
- 22 Tim Jacobs and Gopal Pandurangan. Stochastic analysis of a churn-tolerant structured peer-to-peer scheme. *Peer-to-Peer Networking and Applications*, 6(1):1–14, 2013.

- 23 David R. Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 741–750. ACM, 2002. doi:10.1145/509907.510013.
- 24 Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, 2000.
- 25 Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Proceedings of the 11th Canadian Conference on Computational Geometry, UBC, Vancouver, British Columbia, Canada, August 15-18, 1999*, 1999. URL: <http://www.cccg.ca/proceedings/1999/c46.pdf>.
- 26 Mike Krebs and Anthony Shaheen. *Expander families and Cayley graphs: a beginner's guide*. Oxford University Press, 2011.
- 27 Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72, 2003.
- 28 C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *IEEE INFOCOM*, pages 2133–2143, 2003.
- 29 Peter Mählmann and Christian Schindelhauer. Distributed random digraph transformations for peer-to-peer networks. In *Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 308–317, 2006.
- 30 Yifan Mao, Soubhik Deb, Shaileshh Bojja Venkatakrishnan, Sreeram Kannan, and Kannan Srinivasan. Perigee: Efficient peer-to-peer network design for blockchains. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 428–437, 2020.
- 31 Andrew K. Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin 's public topology and influential nodes, 2015.
- 32 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2nd edition, 2017.
- 33 S. Muthukrishnan and Gopal Pandurangan. Thresholding random geometric graph properties motivated by ad hoc sensor networks. *J. Comput. Syst. Sci.*, 76(7):686–696, 2010.
- 34 Edgar M. Palmer. *Graphical Evolution: An Introduction to the Theory of Random Graphs*. John Wiley & Sons, Inc., USA, 1985.
- 35 Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Building low-diameter P2P networks. In *IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 492–499, 2001.
- 36 Gopal Pandurangan, Peter Robinson, and Amitabh Trehan. Dex: self-healing expanders. *Distributed Computing*, 29(3):163–185, 2016.
- 37 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000.
- 38 Mathew D. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.
- 39 C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory Comput. Syst.*, 32(3):241–280, 1999. doi:10.1007/s002240000118.
- 40 Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001: IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany, November 12–16, 2001 Proceedings 2*, pages 329–350. Springer, 2001.
- 41 Atish Das Sarma, Anisur Rahaman Molla, and Gopal Pandurangan. Distributed computation in dynamic networks via random walks. *Theor. Comput. Sci.*, 581:45–66, 2015. doi:10.1016/j.tcs.2015.02.044.
- 42 Atish Das Sarma, Danupon Nanongkai, Gopal Pandurangan, and Prasad Tetali. Distributed random walks. *J. ACM*, 60(1):2:1–2:31, 2013. doi:10.1145/2432622.2432624.