# Overlay Network Construction: Improved Overall and Node-Wise Message Complexity

**Yi-Jun Chang**
National University of Singapore, Singapore

**Yanyu Chen**
National University of Singapore, Singapore

**Gopinath Mishra**
National University of Singapore, Singapore

## Abstract

We consider the problem of constructing distributed overlay networks, where nodes in a reconfigurable system can create or sever connections with nodes whose identifiers they know. Initially, each node knows only its own and its neighbors' identifiers, forming a local channel, while the evolving structure is termed the global channel. The goal is to reconfigure any connected graph into a desired topology, such as a bounded-degree expander graph or a well-formed tree (WFT) with a constant maximum degree and logarithmic diameter, minimizing the total number of rounds and message complexity. This problem mirrors real-world peer-to-peer network construction, where creating robust and efficient systems is desired.

We study the overlay reconstruction problem in a network of $n$ nodes in two models: GOSSIP-reply and HYBRID. In the GOSSIP-reply model, each node can send a message and receive a corresponding reply message in one round. In the HYBRID model, a node can send $O(1)$ messages to each neighbor in the local channel and a total of $O(\log n)$ messages in the global channel.

In both models, we propose protocols for WFT construction with $O\left(n \log n\right)$ message complexities using messages of $O(\log n)$ bits. In the GOSSIP-reply model, our protocol takes $O(\log n)$ rounds while in the HYBRID model, our protocol takes $O(\log^2 n)$ rounds. Both protocols use $O\left(n \log^2 n\right)$ bits of communication.

We obtain improved bounds over prior work:

**GOSSIP-reply:** A recent result by Dufoulon et al. (ITCS 2024) achieved $O\left(\log^5 n\right)$ round complexity and $O\left(n \log^5 n\right)$ message complexity using messages of at least $\Omega\left(\log^2 n\right)$ bits in GOSSIP-reply. With messages of size $O\left(\log n\right)$, our protocol achieves an optimal round complexity of $O(\log n)$ and an improved message complexity of $O(n \log n)$.

**HYBRID:** Götte et al. (Distributed Computing 2023) showed an optimal $O(\log n)$-round algorithm with $O(\log^2 n)$ global messages per round which incurs a message complexity of $\Omega(m)$, where $m$ is the number of edges in the initial topology. At the cost of increasing the round complexity to $O(\log^2 n)$ while using only $O(\log n)$ messages globally, our protocol achieves a message complexity that is independent of $m$. Our approach ensures that the total number of messages for node $v$, with degree $\deg(v)$ in the initial topology, is bounded by $O\left(\deg(v) + \log n\right)$, while the algorithm of Götte et al. requires $O\left(\deg(v) + \frac{\log^4 n}{\log\log n}\right)$ messages per node.

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms

**Keywords and phrases** Distributed algorithms, Overlay networks, Expander graphs

**Digital Object Identifier** 10.4230/LIPIcs.FSTTCS.2025.21

**Related Version** *Full Version*: https://arxiv.org/abs/2412.04771 [14]

**Funding** The research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (24-1323-A0001). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the Ministry of Education, Singapore.

45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2025).
Editors: C. Aiswarya, Ruta Mehta, and Subhajit Roy; Article No. 21; pp. 21:1–21:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Acknowledgements** We are grateful to an anonymous FSTTCS reviewer for suggesting the use of the hash functions of [31] to reduce the number of bits required for identifying an outgoing edge. In an earlier version of this work, this task had been carried out using the linear sketches of [1]. The incorporation of the reviewer's suggestion results in an $O(\log n)$-factor improvement in the overall communication complexity.

# 1 Introduction

Many of today's large-scale distributed systems on the Internet, such as peer-to-peer (P2P) and overlay networks, prioritize forming logical networks over relying (only) on the physical infrastructure of the underlying network. In these systems, direct connections between nodes can be virtual, using the physical connections of the Internet, and nodes are considered connected if they know each other's IP addresses, allowing them to communicate and establish links. Examples of such systems include cryptocurrencies, the Internet of Things, the Tor network, and overlay networks like Chord [43], Pastry [42], and skip graphs [4]. These networks have the flexibility to reconfigure themselves by choosing which connections to establish or drop. This work focuses on the challenge of efficiently constructing a desired overlay network from any starting configuration of $n$ nodes, recognizing that the problem has a lower bound of $O(\log n)$, since even in an optimal scenario, it takes at least $O(\log n)$ rounds for two endpoints to connect if the nodes initially form a line.

In this paper, we address the well-explored challenge of efficiently constructing overlay topologies in a distributed manner within reconfigurable networks. This task is crucial in modern P2P networks, where topological properties are vital in ensuring optimal performance. Over the past two decades, numerous theoretical studies [39, 34, 23, 17, 29, 10], have focused on developing P2P networks that exhibit desirable characteristics such as high conductance, low diameter, and resilience to substantial adversarial deletions. The common approach in these studies is to build a bounded-degree random graph topology in a distributed way, which ensures these properties. This approach leverages the fact that random graphs are likely to be expanders, possessing all the desired attributes [37, 27]. Random graphs have been extensively used to model P2P networks [39, 34, 37, 35, 17, 11, 8, 7, 9], and the random connectivity topology has been widely adopted in many contemporary P2P systems, including those underpinning blockchains and cryptocurrencies like Bitcoin [36].

Several works have focused on overlay construction that transforms an arbitrary connected graph into a desired topology [3, 24, 25, 26, 19]. While minimizing the number of rounds is the primary objective, reducing the total number of messages exchanged (message complexity) is also crucial. The message complexity in [19] is $O(n \log^5 n)$, whereas in other works it is $\Omega(m)$, where $n$ is the number of nodes and $m$ is the number of edges in the initial topology. In this work, we propose protocols in two models, GOSSIP-reply and HYBRID (defined formally later), that are both round-efficient and communication-efficient. Additionally, our protocol in the HYBRID model optimizes the node-wise message complexity (i.e., the number of messages each node sends and receives based on its degree) compared to the previous work [26].

Before discussing our results and comparing them with previous works, we formally introduce the models in the next section.

## 1.1 Models

We consider synchronous models on a fixed set of nodes $V$, where each node $v \in V$ has a unique identifier $\mathrm{id}(v)$ of length $O(\log n)$, with $n = |V|$. The *local/input* network is represented by a graph $G = (V, E)$. Without loss of generality, we assume that $G$ is connected. Computation proceeds in synchronous rounds, during which the *global/overlay* network evolves. In each round, nodes can send and receive messages and perform local computations. Unless otherwise specified, messages are $O(\log n)$ bits.

The network is *reconfigurable* in the sense that if $u$ knows the identifier of $v$, then $u$ can send a reconfiguration message to $v$ to establish or drop the communication link. Lastly, we also allow implicit edge deletion since it can be easily implemented by only keeping edges established after round $r$.

In general, we assume that the communication links are reliable, and no messages are dropped when the message capacity of the link is not exceeded. We also assume that there is sufficient memory on each computing node for our algorithm to run correctly and is capable of processing all incoming messages at the start of a round within that same round.

In this paper, as already mentioned, we consider two synchronous models: GOSSIP-reply and HYBRID, formally defined as follows.

**GOSSIP-reply model.** One of the earliest works in overlay construction by Angluin et al. [3] considered a model – now known as the GOSSIP model – where each node is allowed to send only one message per round. Recently, a reply version of this model, the GOSSIP-reply $(b)$ model, was introduced by [19], where they developed the first communication-efficient protocol.[1] In this model, each node $v$ can perform the following actions in one round:

1. Send a message of $O(b)$ bits to a neighbor, where any node whose identifier is known to $v$ is considered a neighbor of $v$. We call this message the *contacting message*.
2. Receive all messages sent to $v$. Do some local computation.[2]
3. Send an $O(b)$-bit reply to each of the contacting messages.
4. Receive an $O(b)$-bit reply. Do some local computation.

Observe that in this model, there will be at most $2n$ messages sent in each round, $n$ contacting messages and $n$ replying messages. Hence, any algorithm with $O(T)$ round complexity has $O(nT)$ message complexity.

**HYBRID model.** The hybrid model was proposed in [6] to study shortest path problems and later considered by [26] in the context of overlay construction problem. In this model, the communication is done over both local and global channels. The HYBRID $(\alpha, \beta, \gamma)$ model is defined by three parameters $\alpha$, $\beta$, and $\gamma$:

- Each message size is $O(\alpha)$ bits.
- Each node can send and receive $O(\beta)$ messages per round to each local neighbor, i.e., the local capacity is $O(\beta)$.
- Each node can send and receive $O(\gamma)$ messages per round to any node whose identifier it knows, i.e., the global capacity is $O(\gamma)$.[3]

A subtle aspect of the model arises when a node is sent more messages in a round than its capacity permits. A standard assumption is that the node receives an arbitrary subset of these messages while the rest are dropped. All our algorithms guarantee that, with high probability, this situation never occurs, so the exact handling of such cases is irrelevant to our results.

[1] In [19], it is referred to simply as the GOSSIP-based model or the P2P-GOSSIP model.

[2] Although local computation in Step 2 is not explicitly included in the original definition of [19], preparing the outgoing messages appears to require some. Nevertheless, two rounds of the model without local computation in Step 2 suffice to simulate one round of the model with it.

[3] All $O(\gamma)$ messages can be directed to a single global neighbor or spread across $O(\gamma)$ global neighbors.

In this work, we study the HYBRID$(\log n, 1, \log n)$ model, where each message has size $O(\log n)$ bits, the local capacity is $O(1)$, and the global capacity is $O(\log n)$. Importantly, a node may send a number of local messages proportional to its degree, whereas in the global network it is limited to $O(\log n)$ messages. The rationale for assigning different capacities lies in the cost assumption: local communications, which take place on the given topology, are cheaper, while global communications, which require establishing new links, are more costly.

In addition to the GOSSIP-reply and HYBRID models, the P2P-CONGEST model has also been widely studied [24, 25, 26]. In P2P-CONGEST, each node can send and receive up to $O(\Delta \log n)$ messages of $O(\log n)$ bits per round, where $\Delta$ is the maximum degree of any node in the initial topology. It is important to note that HYBRID $(\log n, 1, \log n)$ is *weaker* than P2P-CONGEST in the sense that any protocol in HYBRID $(\log n, 1, \log n)$ can be simulated in the P2P-CONGEST model without asymptotically increasing the round or message complexity asymptotically. Therefore, all results obtained in HYBRID $(\log n, 1, \log n)$ also apply to P2P-CONGEST.

## 1.2 Our contribution and comparison with prior work

As already mentioned, we focus on designing algorithms in both the GOSSIP-reply and HYBRID models that are efficient in terms of rounds and communication. Additionally, we demonstrate that our protocol for the HYBRID model also achieves improved node-wise message complexity. We also discuss implications for the P2P-CONGEST model. See Appendix B for a discussion on the tradeoffs between different complexity measures and the motivation for studying node-wise message complexity.

Unless otherwise specified explicitly, all of our results including prior works are randomized and succeed *with high probability* (w.h.p.), i.e., with probability at least $1 - 1/\text{poly}(n)$.

### 1.2.1 Our result in the GOSSIP-reply model

Our result in the GOSSIP-reply model is summarized in the following theorem. It shows that starting from any arbitrary topology, we can transform it into a star overlay. We then demonstrate how a star overlay can be converted into a desired topology by leveraging the properties of the GOSSIP-reply model.

▶ **Theorem 1.1.** *There is a protocol in the* GOSSIP-reply $(b)$ *model that can construct a star overlay in* $O\left(\log n \cdot \max\left(\frac{\log n}{b}, 1\right)\right)$ *rounds with* $O\left(n \log n \cdot \max\left(\frac{\log n}{b}, 1\right)\right)$ *messages w.h.p.*

Note that building a star topology is similar to doing leader election in the *reconfigurable* network. Observe that when the star topology is constructed, we can treat the distinguished center node in the star as the leader and perform many tasks on the leader node locally. More specifically, in GOSSIP-reply$(b)$, we can reconfigure the network from the star topology to any topology whose maximum degree is $\Delta = O\left(\frac{b}{\log n}\right)$ in $O(1)$ round.

▶ **Observation 1.2.** *If the initial topology $G$ is a star, then there is an $O(\Delta(H))$-round protocol in the* GOSSIP-reply $(b)$ *model to construct an overlay network with a desired topology $H$ whose maximum degree is $\Delta(H) = O\left(\frac{b}{\log n}\right)$.*

**Proof.** Firstly, every node except the distinguished center node sends a request message to the center node. Then the center node will compute an assignment of the nodes in the desired topology locally and reply to every node $v$ with their neighborhood $N_H(v)$. Each node then takes $O(\Delta(H))$ rounds to establish connections with the new neighbors formally. Note that the center node needs to send $O\left(\Delta(H)\log n\right)$ bits to every leaf node. Since $\Delta(H) = O\left(\frac{b}{\log n}\right)$, the information that the center node needs to send is $O\left(\Delta(H)\log n\right) = O(b)$ bits, which can be sent in one message. ◀

We obtain the following corollary by applying Observation 1.2 and Theorem 1.1 with $b = O(\log n)$.

▶ **Corollary 1.3.** *There is a protocol in the* GOSSIP-reply $(\log n)$ *model that can construct any constant degree overlay network in* $O(\log n)$ *rounds with* $O(n\log n)$ *messages w.h.p.*

**Comparison with Dufoulon et al. [19].** The algorithm by Dufoulon et al. in the GOSSIP-reply $(b)$ model, with $b = \Omega(\log^2 n)$, converts any arbitrary topology into a constant-degree expander in $O(\log^5 n)$ rounds, with a message complexity of $O(n\log^5 n)$. Thus, Corollary 1.3 provides a strict improvement over [19] in both round and message complexity. Additionally, our algorithm can produce any constant-degree overlay, whereas the algorithm of [19] could only construct a constant-degree expander. The comparison of our result in the GOSSIP-reply $(b)$ model with that of [19] is also presented in Table 1.

**Table 1** Improvements in the GOSSIP-reply $(b)$ model.

| Reference | $b$ | Rounds | Total message complexity | Target topology |
|---|---|---|---|---|
| [19] | $\Omega(\log^2 n)$ | $O(\log^5 n)$ | $O(n\log^5 n)$ | $O(1)$-degree expander |
| Corollary 1.3 | $O(\log n)$ | $O(\log n)$ | $O(n\log n)$ | Any $O(1)$-degree graph |

### 1.2.2 Our result in the HYBRID model

In the HYBRID $(\log n, 1, \log n)$ model, our main result is summarized in the following theorem: we show that starting from any arbitrary initial topology, it is possible to transform the network into a *well-formed tree* (WFT) efficiently. A well-formed tree with $n$ nodes is defined as one that has a constant maximum degree and a depth of $O(\log n)$. Furthermore, we discuss how a well-formed tree can be efficiently converted into a constant-degree expander in HYBRID$(\log n, 1, \log n)$ model using the results from prior work in [19, 26].

▶ **Theorem 1.4.** *There is a protocol in the* HYBRID $(\log n, 1, \log n)$ *model that can construct a well-formed tree overlay from any input graph* $G$ *in* $O\left(\log^2 n\right)$ *rounds with* $O\left(n\log n\right)$ *messages w.h.p. Moreover, each node* $v$ *sends and receives at most* $O(\deg_G(v) + \log n)$ *messages throughout the protocol.*

We remark that, although the sum of the node-wise bounds appears to imply an $O(m)$ message complexity, in our algorithm only a small subset of nodes may incur as many as $\Omega(\deg_G(v))$ messages, and the message complexity remains bounded by $O(n\log n)$. Due to the use of randomness, it is not possible to determine in advance which nodes incur these higher costs.

Our algorithm follows a Boruvka-style cluster-merging process while maintaining the invariant that each cluster induces a well-formed tree. Outgoing edges are identified using sketching techniques. To achieve the node-wise message bound of $O(\deg_G(v) + \log n)$, we address the high communication load on star centers during cluster merges by introducing a matching-based method that pairs clusters for merging, even without a direct connecting edge, while ensuring that the total number of clusters reduces by a constant factor in each round of the merging process.

To further optimize the message complexity, we employ a randomized procedure for constructing an $O(\log n)$-degree, $O(\log n)$-depth tree from a cycle. This improves upon the deterministic pointer-jumping process of prior work, yielding an $O(\log n)$-factor reduction in message cost.

Through minor modifications of the works in [19, 26], we restate the following lemma, which enables the efficient transformation of a constant degree overlay network (e.g., a well-formed tree) into a constant degree expander network with constant conductance w.h.p. in the $\mathsf{HYBRID}(\log n, 1, \log n)$ model.

▶ **Lemma 1.5** ([19, 26]). *Consider the $\mathsf{HYBRID}(\log n, 1, \log n)$ model. For any constant $\Phi \in (0, 1/10]$, there is a protocol that takes $O\left(\log^2 n\right)$ rounds and $O\left(n \log^2 n\right)$ messages to convert an $O(1)$-degree overlay network into an $O(1)$-degree expander network with conductance at least $\Phi$, w.h.p. Moreover, each node $v$ sends and receives at most $O\left(\frac{\log^3 n}{\log \log n}\right)$ messages w.h.p.*

By applying Lemma 1.5 after Theorem 1.4, we can construct an expander overlay network in the $\mathsf{HYBRID}(\log n, 1, \log n)$ model in $O(\log^2 n)$ rounds with $O(n \log^2 n)$ messages.

▶ **Corollary 1.6.** *There is a protocol in the $\mathsf{HYBRID}(\log n, 1, \log n)$ model with the following guarantees:*

- *It constructs a constant-degree expander graph from any input graph $G$ in $O\left(\log^2 n\right)$ rounds using $O\left(n \log^2 n\right)$ messages w.h.p.*
- *Each node $v$ sends and receives at most $O\left(\deg_G(v) + \frac{\log^3 n}{\log \log n}\right)$ messages.*

**Comparison with Götte et al. [26].** Götte et al. investigated the problem of overlay reconstruction in the $\mathsf{HYBRID}(\log n, 1, \log^2 n)$ model, aiming to convert an arbitrary initial topology into a well-formed tree or a constant-degree expander. Their algorithm achieved an optimal round complexity of $O(\log n)$ rounds with a message complexity of $\Omega(m + n \log^3 n)$, and the maximum number of messages sent or received by a node of degree $\deg_G(v)$ is $O\left(\deg_G(v) + \frac{\log^4 n}{\log \log n}\right)$. In comparison, although our result in Theorem 1.4 and Corollary 1.6 require $O(\log^2 n)$ rounds, it operates in the weaker $\mathsf{HYBRID}(\log n, 1, \log n)$ model. Crucially, the message complexity of our algorithm does not depend on $m$, and it achieves better node-wise message complexity compared to [26]. It is important to note, however, that our approach does not lead to an $O(\log n)$-round algorithm even in the $\mathsf{HYBRID}(\log n, 1, \log^2 n)$ model. The comparison of our result in the $\mathsf{HYBRID}$ model with that of [26] is presented in Table 2. An open question remains: is it possible to achieve the optimal $O(\log n)$ rounds in the $\mathsf{HYBRID}(\log n, 1, \log n)$ model (or even in the $\mathsf{HYBRID}(\log n, 1, \log^2 n)$ model) with a message complexity of $O(n \cdot \text{poly}(\log n))$?

**Table 2** Improvements in the $\mathsf{HYBRID}\,(\log n, 1, \gamma)$ model.

| Reference | $\gamma$ | Rounds | Message complexity | | Target topology |
|---|---|---|---|---|---|
| | | | Total | Node-wise | |
| Götte et al. [26] | $O(\log^2 n)$ | $O(\log n)$ | $\Omega\big(m + n \log^3 n\big)$ | $O\big(\deg_G(v) + \frac{\log^4 n}{\log \log n}\big)$ | WFT / $O(1)$-degree expander |
| Theorem 1.4 | $O(\log n)$ | $O(\log^2 n)$ | $O(n \log^2 n)$ | $O\big(\deg_G(v) + \log^2 n\big)$ | WFT |
| Corollary 1.6 | $O(\log n)$ | $O(\log^2 n)$ | $O(n \log^2 n)$ | $O\big(\deg_G(v) + \frac{\log^3 n}{\log \log n}\big)$ | $O(1)$-degree expander |

### 1.2.3 Implication in P2P-CONGEST model

Götte et al. [26] considered two models: P2P-CONGEST and HYBRID $\left(\log n, 1, \log^2 n\right)$, which are not directly comparable. In particular, the result of Götte et al. [26] in HYBRID $\left(\log n, 1, \log^2 n\right)$ does not translate directly to P2P-CONGEST. However, as already mentioned before, any protocol in HYBRID $(\log n, 1, \log n)$ can be simulated in the P2P-CONGEST model without asymptotically increasing the round or message complexity. Therefore, from Theorem 1.4 and Corollary 1.6, we obtain the following corollary.

▶ **Corollary 1.7.** *There is a protocol in the* P2P-CONGEST *model that can construct a well-formed tree or a constant-degree expander graph from any input graph $G$ in $O\left(\log^2 n\right)$ rounds. The algorithm uses $O\left(n \log n\right)$ messages or $O\left(n \log^2 n\right)$ w.h.p. for well-formed tree or a constant-degree expander graph, respectively. Moreover, each node $v$ sends and receives at most $O(\deg_G(v) + \log n)$ or $O\left(\deg_G(v) + \frac{\log^3 n}{\log\log n}\right)$ messages depending on whether the target topology is a well-formed tree or a constant-degree expander graph, respectively.*

**Comparison with Götte et al. [26].** Götte et al. studied the overlay reconstruction problem in the P2P-CONGEST model, where the goal was to transform an arbitrary initial topology into a well-formed tree. Their solution achieved an optimal round complexity of $O(\log n)$ with a message complexity of $\Omega(m \log n + n \log^2 n)$, and the maximum number of messages sent or received by a node of degree $\deg_G(v)$ is $O\left(\deg_G(v) \cdot \frac{\log^3 n}{\log\log n}\right)$. In contrast, our algorithm, as stated in Corollary 1.7, requires $O(\log^2 n)$ rounds but notably achieves message complexity independent of $m$ and improved node-wise message complexity. The comparison of our implication in the P2P-CONGEST model with that of the result of [26] is also presented in Table 3. A key remaining open question is whether it is possible to attain the optimal $O(\log n)$ rounds in the P2P-CONGEST model with a message complexity of $O(n \cdot \text{poly}(\log n))$.

**Table 3** Improvements in P2P-CONGEST model.

| Reference | Rounds | Message complexity | | Target topology |
|---|---|---|---|---|
| | | Total | Node-wise | |
| [26] | $O(\log n)$ | $\Theta(m \log^2 n)$ | $O\left(\deg_G(v) \cdot \frac{\log^3 n}{\log\log n}\right)$ | WFT/ $O(1)$-degree expander |
| Corollary 1.7 | $O(\log^2 n)$ | $O(n \log n)$ | $O\left(\deg_G(v) + \log n\right)$ | WFT |
| Corollary 1.7 | $O(\log^2 n)$ | $O(n \log^2 n)$ | $O\left(\deg_G(v) + \frac{\log^3 n}{\log\log n}\right)$ | $O(1)$-degree expander |

## 1.3 Number of bits communicated

For all of Theorem 1.1, Corollary 1.3, Theorem 1.4, and Corollary 1.6, the total number of bits communicated among the nodes in all protocols is $O(n \log^2 n)$, due to the use of the hashing techniques from King King, Kutten, and Thorup [31]. This bound on the communication complexity breaks the $\Omega(n \log^3 n)$ barrier of the linear sketch of Ahn, Guha, and McGregor [1] which was used in the previous work [19].

It has been shown that $\Omega(n \log^3 n)$ bits of communication are indeed necessary for several applications of the linear sketch of Ahn, Guha, and McGregor [1], such as distributed and sketching spanning forest [38] and connectivity [44]. More concretely, in the distributed sketching model, the goal of the connectivity problem is to determine whether an $n$-node graph $G$ is connected, with each of the $n$ players having access to the neighborhood of a single vertex. Each player sends a message to a central referee, who then decides whether $G$ is connected. Yu [44] established that for the referee to decide correctly with probability 1/4, the total communication must be at least $\Omega(n \log^3 n)$ bits.

The ability to break this barrier stems from being able to communicate in both ways in multiple rounds as opposed to the one-round one-way setting described above. The optimality of the hashing technique from King, Kutten, and Thorup [31] is much less well-known, and it remains open whether their communication complexity bound can be further improved. Any such improvement will likely lead to improvements in the $O(n \log^2 n)$ communication complexity bound in this paper as well as many other applications of the hashing technique.

### 1.4 Organization

In Section 2, we present the basic graph terminologies and tools. In Section 3, we present our protocol in the GOSSIP-reply model, proving Theorem 1.1. In Section 4, we present our protocol in the HYBRID model, proving Theorem 1.4. In Appendix A, we present a comprehensive discussion of related work. In Appendix B, we discuss the tradeoffs between some complexity measures. For any missing details, please refer to the full version of the paper [14].

## 2 Preliminaries

A graph is defined as $G = (V, E)$, where $E \subseteq \binom{V}{2}$, as the edges are undirected. The graph does not allow self-loops or multi-edges. Let $n = |V|$ and $m = |E|$. The neighborhood of a vertex $v$ in $G$ is denoted as $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$, and the degree of a vertex $v$ in $G$ is defined as $\deg_G(v) := |N_G(v)|$. The maximum degree of the graph is represented as $\Delta(G) = \max_{v \in V} \deg_G(v)$. The distance $d(u, v)$ between any two nodes $u$ and $v$ is the number of edges in the shortest path between them. The diameter of a graph is the maximum distance between any two nodes in the graph. The set of connected components of $G$ is denoted as $\mathsf{CC}(G)$, and the number of connected components in $G$ is represented by $\mathsf{cc}(G)$.

A star graph, denoted as $K_{1,n-1} = (V, E)$, has a distinguished node $v \in V$ such that an edge $e = \{u, w\} \in E$ exists if and only if $v \in e$. A tree $T$ is a connected acyclic graph. A rooted tree $T_v$ is a tree with a distinguished node $v$ serving as the root. The depth of a rooted tree $T_v$ is the maximum distance from the root $v$ to any other node in the tree. A *well-formed tree* is defined as a rooted tree with a constant maximum degree and a depth of $O(\log n)$. A *satisfactory tree* is defined as a rooted tree with $O(\log n)$ maximum degree and $O\left(\log n\right)$ depth.

We assume that each node has an ID of length $O(\log n)$ bits. The ID of an edge is a concatenation of the node IDs with the smaller first. We use $\eta(T_x)$ to denote the maximum edge ID among all edges incident to nodes in $T_x$.

**Expander.** The volume of any subset $S \subseteq V$ is defined as $\mathrm{vol}(S) := \sum_{v \in S} \deg_G(v)$. The conductance of a subset $S \subseteq V$, where $|S| \neq 0$ and $|S| \neq |V|$, is given by

$$\Phi_G(S) := \frac{|E(S, V \setminus S)|}{\min(\mathrm{vol}(S), \mathrm{vol}(V \setminus S))},$$

where $E(S, V \setminus S) := \{\{u, v\} \in E \mid u \in S, v \in V \setminus S\}$ represents the set of edges between $S$ and its complement.

The conductance of the graph $G$ is defined as

$$\Phi(G) := \min_{S \subseteq V, S \neq \emptyset, S \neq V} \Phi_G(S).$$

Informally, a graph is considered an expander if it has high conductance. The thresholds commonly used to define high conductance vary by context, including $1/n^{o(1)}$, $1/\operatorname{polylog}(n)$, and $1/O(1)$. In this paper, we define an expander as a graph with conductance of $1/O(1)$.

**Broadcast-and-echo.** A basic distributed protocol to disseminate and gather information is *broadcast-and-echo*. It is initiated by some node $x$ and messages are relayed in a BFS manner, with possible modifications to the messages down the broadcasting tree. Then this process reaches the leaves, leaf nodes echo with some messages back to their parents. Internal nodes wait untill all the messages are gathered before sending a computed aggregated message to their parents. This process takes $O(D(T_x))$ rounds and $O(|T_x|)$ messages, where $T_x$ refers to the broadcasting tree in this process.

More generally, in the CONGEST model with bandwidth $B$ bits, a broadcast-and-echo initiated by $x$ in $T_x$ with a maximum message size of $S$ bits can be done in $O\left(\frac{S}{B} + D(T_x)\right)$ rounds and $O\left(\frac{S}{B}|T_x|\right)$ messages, via message pipelining.

**Find any outgoing edge.** For any tree $T_x$ rooted at $x$, we call edges between $T_x$ and $V \setminus T_x$ outgoing. Linear sketch techniques used in previous works by [1, 30, 40, 19] of $O(\log^2 n)$ bits can be used to sample an outgoing edge with constant success probability. To save on message complexity, we instead use a subroutine from [31] to find an arbitrary outgoing edge from $T_x$.

At a high level, this protocol of [31] uses similar observation to the well-known linear graph sketch that internal edges in a tree will contribute 0 to the sum of degree, or XOR of edge IDs. However, it breaks the well-known linear graph sketch [1] into two phases. First, it uses $O(\log n)$ bits to aggregate the parity of the number of edges that is hashed into each log-scale bracket (1,2,4,8, ...). Then, they show that with constant probability there is one log-scale bracket that has exactly one edge hashed to it. This step corresponds to guessing the suitable sampling probability for exactly one outgoing edge to be sampled. They finally spend another $O(\log n)$ bits to identify the identity of the edge by XORing the edge numbers that are in the identified bracket. This process takes four iterations of broadcast-and-echo with message size $O(\log n)$ bits.

For completeness, we describe the protocol FindOutgoing(x) initiated at node $x$, which returns an edge leaving $T_x$ with probability at least 1/16. The version described here corresponds to FindAny-C(x) in [31]. FindOutgoing(x) uses another protocol from [31] HPTestOut(x) that returns true with high probability if there is an edge leaving $T_x$ and false otherwise. HPTestOut is always correct if true is returned and uses one broadcast-and-echo with message size $O(\log n)$ bits.

FindOutgoing($x$):

1. $x$ initiates HPTestOut($x$) in $T_x$ and return $\emptyset$ if HPTestOut returns false.
2. Determine the identity of an edge with the following steps:
   a. $x$ broadcasts a random pairwise independent hash function $h : [1, \eta(T_x)] \to [0, r]$ where $r = 2^w > \sum_{v \in T_x} \deg(v)$ for some $w$ .
   b. each node $y$ hashes the ID of all edges incident to it and compute a $\log r$-bit binary vector $\vec{h}(y)$ such that $\vec{h}_i(y) := |\{e \mid y \in e \wedge h(e) < 2^i\}| \mod 2$.

   **c.** The vector $\vec{h}(T) := \oplus_{y \in T} \vec{h}(y)$ is computed up the tree, in the broadcast-and-echo return to $x$. Then $x$ broadcasts $min = \min\{i \mid \vec{h}_i(T) = 1\}$.
   **d.** Each node $y$ computes $s(y) = \oplus\{e \mid y \in e \wedge h(e) < 2^{min}\}$ and $s(T) = \oplus_{y \in T} s(y)$ is computed up the tree in the broadcast-and-echo and returned to $x$. Observe that if there is exactly one edge leaving $T_x$ with $h(e) < 2^{min}$, then $s(T)$ is its edge ID.

**3.** $x$ can perform another broadcast-and-echo to check if $s(T_x)$ is indeed a valid edge ID and return $s(T)$ if the check succeeds and $\emptyset$ if the check fails.

▶ **Lemma 2.1** ([31]). *If there is no edge leaving $T_x$, then* FINDOUTGOING*(x) return $\emptyset$. Otherwise,* FINDOUTGOING$(x)$ *returns an edge leaving $T_x$ with probability at least $1/16$, else it returns $\emptyset$. The algorithm uses worst-case $O(D(T_x))$ rounds and $O(|T_x|)$ messages.*

## 3 Star overlay construction in the GOSSIP-reply model

We begin by describing the high-level approach underlying our algorithm for the GOSSIP-reply model. We draw inspiration from the following existing techniques.

**Boruvka-style cluster merging with efficient inter-cluster edge selection.** We use a Boruvka-style cluster merging approach used in many prior works [19, 24, 3] while maintaining a simple and useful invariant. We start with each node being a single cluster. In each iteration, we select inter-cluster edges and merge the clusters joined by these edges. By ensuring a constant factor reduction in the number of clusters in each iteration, the process terminates in $O(\log n)$ iterations. The challenge here is how to quickly select an outgoing edge effectively (effectiveness measured by small round or message complexity). We adopt Lemma 2.1 to find an outgoing edge efficiently. This was not previously used in the overlay network construction context and is more efficient than the linear graph sketching technique used by [19].

**Selective merging to overcome long chains.** Overall our protocol works by sampling an inter-cluster edge from each cluster and merging clusters joined by sampled edges. Merging can be potentially slow due to the large diameter when the inter-cluster edges form a long chain connecting many clusters. Therefore, we break this chain via a simple coin-flipping technique, where each cluster flips a coin and will only accept a request if the coins of the requesting and requested clusters satisfy a specific condition. This symmetry-breaking technique is used extensively in many parallel and distributed works in problems such as parallel list ranking [16] and distributed graph connectivity [21].

**Faster and simpler merging.** A key difference between our protocol and that of [19] is that we maintain a much simpler and useful invariant (maintaining a star in each cluster) that allows us to aggregate information in a cluster and perform merging among clusters much faster.

**Star overlay construction protocol.** We describe our protocol MERGESTAR to construct a star overlay in the GOSSIP-reply $(\log n)$ model which proves Theorem 1.1.

The algorithm proceeds in $O(\log n)$ Boruvka-style phases w.h.p. In each phase, a constant factor of clusters is merged into other clusters to form a cluster for the next phase with constant probability. The algorithm starts with each node being a cluster and maintains the following invariant: at the end of each phase, every node in each cluster $S$ agrees on a leader $l(S)$. This is true initially since each node can be the leader of its own cluster. In other words, this invariant implies that each cluster will keep a star overlay topology.

Denote the given input topology as $G = (V, E)$. Let $G_i = (V, E_i)$ be the topology of the overlay network at the end of phase $i$. Let $G_0 = (V, \emptyset)$, i.e., we start with each node being an isolated node in the overlay network. We refer to a connected component $C = \left(S, E_i \cap \binom{S}{2}\right) \in \mathsf{CC}(G_i)$ as a cluster in $G_i$.[4] An outgoing edge from the cluster $C$ is an edge in the input graph $G = (V, E)$ connecting a node in $S$ to a node outside $S$ i.e., $\mathrm{Out}(C) := E_G(S, V\backslash S) := \{\{u, v\} \in E \mid u \in S \text{ and } v \in V\backslash S\}$ is the set of outgoing edges of the cluster $C$.

Each phase consists of three steps. In phase $i$, we start with the overlay $G_{i-1}$.

1. *Sampling Step:* Each cluster $C = \left(S, E_i \cap \binom{S}{2}\right)$ finds an outgoing edge $e$ from $\mathrm{Out}(C)$ and a color $\chi(S) \in \{\mathsf{Blue}, \mathsf{Red}\}$, and then sends a *merging request* containing the sampled color $\chi(S)$ to the external node in the sampled edge.
2. *Grouping Step:* Clusters who received *merging requests* decide on which clusters to merge with based on the color and reply either with an *accepting message* or a *rejecting message*. The purpose of the $\{\mathsf{Blue}, \mathsf{Red}\}$-coloring is to prevent long chains of *merging requests* slowing down the *Merging Step*.
3. *Merging Step:* Clusters that agree on merging will perform this step to merge the clusters. Each node in these clusters must agree on a new leader to maintain the invariant.

**Sampling step.** In this step, each cluster $C = \left(S, E_i \cap \binom{S}{2}\right)$ needs to sample an outgoing edge uniformly at random with constant probability. It will take $\Omega(\Delta)$ rounds if we let each node check if each neighbor is in the same cluster. We will use the FindOutgoing protocol to reduce communication. Each broadcast-and-echo is replaced by each leave in the star sending one request to the leader $l(S)$ for the broad-casted information. This exploits the replying property of GOSSIP-reply. No random walk or PUSH-style information-spreading like [19] is needed. After the FindOutgoing protocol, the leader finds an outgoing edge with constant probability. Then it sends a *merging request* along the sampled edge to the destination. Note that this step works correctly with probability 1/16 due to Lemma 2.1, as long as $cc(G_i) \geq 2$.

Additionally, to facilitate the grouping step, the leader $l(S)$ independently and uniformly samples a color $\chi(S) \in \{\mathsf{Blue}, \mathsf{Red}\}$ for the cluster $S$ and sends the 1-bit information along with the *merging request*.

**Grouping step.** The node $v \in S'$ that received the *merging request* will reply with the leader $l(S')$. Then each leader will send the request to other cluster leaders. Since each cluster can only initiate one *merging request*, there will be in total $c(G_i)$ merging requests. Thus, there can be cycles or long chains in the graph, which can affect merging speed. Thus, we need to break these chains. We do this by making each cluster leader accept a request if and only if it is Red and the requesting cluster is Blue.

**Merging step.** Note that after the Grouping step, the *merging requests* viewed as edges among clusters will form a star with the Red clusters as the centers. We can identify the merged clusters in $G_{i+1}$ with the Red clusters in $G_i$. Thus, We can maintain the invariant in each cluster in $G_{i+1}$ by letting the leader of the Red clusters become the new leader of the merged clusters in $G_{i+1}$.

[4] Sometimes we also loosely refer to $S$ as the cluster. This should not cause any confusion since a cluster $C$ in $G_i$ is induced by $S$.

Each Red leader will reply with its own identifier to the Blue leaders. Each Blue cluster leader then broadcasts this new leader identifier to the Blue cluster members by replying to the cluster members' request. Actions taken by different nodes in the merging step are summarized in Table 4.

**Table 4** Summary of actions of different nodes in the Merging Step.

| Type in $G_i$ | Actions in the Merging Step |
|---|---|
| Cluster member | Send leader update requests to its leader in $G_i$. |
| Red leader $u$ | Reply to leader update requests with its own identifier. |
| Blue leader $v$ | Received acceptance decision from $u$;<br>$\begin{cases}\text{Reply to leader update requests with id}(u) & \text{if } v \text{ is accepted}\\ \text{Reply to leader update requests with id}(v) & \text{if } v \text{ is rejected}\end{cases}$ |

After this, each node in $G_{i+1}$ will agree on the same leader (i.e., the Red leader), thereby maintaining the invariant.

**Analysis.** We bound the number of phases that the algorithm takes before it terminates w.h.p.

▶ **Lemma 3.1.** *Let $i \in \mathbb{N}$ such that $\mathsf{cc}(G_i) \geq 2$. We have $\mathbb{E}[\mathsf{cc}(G_{i+1})] \leq \frac{63}{64}\mathsf{cc}(G_i)$.*

**Proof.** First, observe that in one phase, each cluster has done the sampling step and the grouping step which can affect the number of clusters at the end of the phase.

Let $X_C$ be the indicator random variable for the event that either $C$ fails to sample an outgoing edge or the *requesting message* initiated by $C$ is **rejected**, for each cluster $C \in \mathsf{CC}(G_i)$. As $cc(G_i) \geq 2$, the leader of each cluster $C \in \mathsf{CC}(G_i)$ find an outgoing edge from $C$ with probability at least $1/16$. Moreover, note that the requesting message from cluster $C$ will be accepted with probability $1/4$. So, we have

$$\mathbb{E}[X_C] \leq 1 - \frac{1}{16}\frac{1}{4} = \frac{63}{64}.$$

By linearity of expectation, we have

$$\mathbb{E}\left[\mathsf{cc}(G_{i+1})\right] = \mathbb{E}\left[\sum_{C \in \mathsf{CC}(G_i)} X_C\right] = \sum_{C \in \mathsf{CC}(G_i)} \mathbb{E}\left[X_C\right] \leq \frac{63}{64}\mathsf{cc}(G_i).$$ ◀

▶ **Lemma 3.2.** *The protocol* MergeStar *terminates in $O(\log n)$ phases w.h.p.*

**Proof.** Let $t = c\log n$ for some sufficiently large constant $c$ that we will determine later. Our goal is to show that $\mathsf{cc}(G_t) = 1$ w.h.p., implying that the protocol terminates in $O(\log n)$ phases w.h.p. Define $Y_i := \mathsf{cc}(G_i) - 1$, where $i$ is a non-negative integer. Initially, we have $Y_0 = n - 1$, and the process terminates in phase $i$ when $Y_i = 0$. We aim to demonstrate that $Y_t = 0$ w.h.p. Observe that $Y_0, \ldots, Y_t$ form a sequence of random variables that take non-negative integer values.

To achieve this, it suffices to show that $\mathbb{E}[Y_i] \leq \frac{63}{64}\mathbb{E}[Y_{i-1}]$ for every $i \in \mathbb{N}$. We start by establishing that $\mathbb{E}[Y_i \mid Y_{i-1}] \leq \frac{63}{64}Y_{i-1}$ for each $i \in \mathbb{N}$.

Consider the case when $Y_{i-1} \geq 1$, i.e., $\mathsf{cc}(G_{i-1}) \geq 2$. Applying Lemma 3.1, we have:

$$\mathbb{E}[Y_i \mid Y_{i-1}] = \mathbb{E}[\mathsf{cc}(G_i) \mid \mathsf{cc}(G_{i-1}) = Y_{i-1} + 1] - 1 \leq \frac{63}{64}(Y_{i-1} + 1) - 1 \leq \frac{63}{64}Y_{i-1}.$$

On the other hand, if $i$ is such that $Y_{i-1} = 0$, then $Y_i = 0$. Thus, $\mathbb{E}[Y_i \mid Y_{i-1}] \leq \frac{63}{64} Y_{i-1}$ holds trivially. Hence, we can conclude:

$$\mathbb{E}[Y_i] = \mathbb{E}[\mathbb{E}[Y_i \mid Y_{i-1}]] \leq \mathbb{E}\left[\frac{63}{64} Y_{i-1}\right] = \frac{63}{64}\mathbb{E}[Y_{i-1}] \quad \text{, for all } i \in \mathbb{N}.$$

This implies:

$$\mathbb{E}[Y_t] \leq \left(\frac{63}{64}\right)^t \mathbb{E}[Y_0] = \left(\frac{63}{64}\right)^t \cdot (n-1).$$

Since $t = c \log n$, we choose $c$ to be sufficiently large such that $\mathbb{E}[Y_t] \leq \frac{1}{\text{poly}(n)}$. Applying Markov's inequality, we have $\Pr[Y_t \geq 1] \leq \frac{1}{\text{poly}(n)}$. Since $Y_t$ only takes non-negative integer values, it follows that $Y_t = 0$ holds w.h.p. Therefore, we conclude that the protocol terminates in $O(\log n)$ phases w.h.p. ◀

Now we are ready to prove Theorem 1.1.

▶ **Theorem 1.1.** *There is a protocol in the* GOSSIP-reply $(b)$ *model that can construct a star overlay in* $O\left(\log n \cdot \max\left(\frac{\log n}{b}, 1\right)\right)$ *rounds with* $O\left(n \log n \cdot \max\left(\frac{\log n}{b}, 1\right)\right)$ *messages w.h.p.*

**Proof.** The correctness of this algorithm is obvious since a leader is maintained and known to all nodes in the clusters after each phase. Once the algorithm terminates, there will be only one cluster and all nodes will agree on a single leader. Therefore, at the end of the algorithm, we have constructed a star overlay network.

By Lemma 3.2, we know that the algorithm terminates in $O(\log n)$ phases w.h.p. Now we only need to check that it takes $O(1)$ rounds in each phase in the GOSSIP-reply $(\log n)$ model to conclude that the algorithm terminates in $O(\log n)$ rounds with high probability. To be exact, we need seven rounds (described from the point of view of a cluster leader): 4 rounds to run FindOutgoing; 1 round to send the *requesting message* and receive the new leader; 1 round to resend the *requesting message* to the cluster leaders and receive an *accepting message* or a *rejecting message*; and 1 round to distribute the new leader identifier to the old cluster members. During this process, every cluster member just keeps sending requests to their old leader for the identifier of the new leader.

For GOSSIP-reply$(b)$, where $b \in O(\log n)$, we can simulate one round of the above process with $O\left(\frac{\log n}{b}\right)$ rounds and arrive at our conclusion. The $O\left(n \log n \cdot \max\left(\frac{\log n}{b}, 1\right)\right)$ messages w.h.p. total message complexity follows from the restriction of the model that at most $O(n)$ messages are sent in each round. ◀

## 4 Well-formed tree overlay construction in the HYBRID model

In this section, we describe a protocol for well-formed tree (WFT) construction in the HYBRID$(\log n, 1, \log n)$ model. Due to Lemma 1.5 adapted from results in [26, 19], we can convert a well-formed tree overlay to a constant-degree expander overlay via only global communications in the HYBRID$(\log n, 1, \log n)$ model with an additive $O(\log^2 n)$ round complexity and additive $O(\log^2 n)$ messages for each node. Therefore, we will focus on describing the WFT construction protocol.

## 4.1 Algorithm

Now we describe our protocol HybridWFT to build the well-formed tree to prove Theorem 1.4. The overall structure of the algorithm is similar to that in Section 3. At the start, each node is a cluster, i.e., $G_0$ consists only of isolated nodes. The algorithm proceeds in phases where in each phase a constant fraction of the clusters are merged in expectation. At the end of $O(\log n)$ phases, there will be only one cluster (per connected component of the input graph) w.h.p. We maintain the following invariant in each cluster after each phase: the subgraph induced by each cluster is a satisfactory tree ($O(\log n)$-degree, $O(\log n)$-depth) that spans the cluster. More specifically, each node in the cluster knows its parent and children, as well as the root of the satisfactory tree.

In each phase, there will be 3 major steps. Unless mentioned otherwise, all communications are done over the global channel.

**Sampling step.** This step aims to sample an outgoing edge from each cluster. We run the protocol FindOutgoing$(r)$ from the root $r$ and find an outgoing edge (from the cluster) with constant probability. This takes $O(D(T_x)) = O(\log |T_x|)$ rounds and $O(|T_x|)$ messages. Next, the root node samples a random color $\chi \in \{\mathsf{Red}, \mathsf{Blue}\}$. Finally, $r$ broadcasts the selected outgoing edge and the color in the cluster.

**Grouping step.** The selected node in each cluster will send a merging request along the selected outgoing edge. The merging request contains the color and the root identifier of the requesting cluster. Since we sample from the local edges, all merging requests will be over the local edges. Then each node receiving a request accepts the request if and only if the accepting cluster is Red and the requesting cluster is Blue. The accepting node sends an accepting message with the identifier of the accepting node via the local edge.

To improve the probability of low local communication, we will need to reduce the effective neighbors of Blue nodes. We do this via the following procedure. Each node $v$ receiving a request will compute a matching over its rejected neighbors. Then $v$ will send the *rejecting message* along with its matched *regrouping cluster* to each rejected requesting node. Then each rejected node will send a *regrouping message* to its matched *regrouping cluster* over the global network. The matched pairs will exchange their cluster leader identifier to decide on a new leader based on who has a larger identifier.

▶ **Lemma 4.1.** *The cluster graph $H^C = (\mathsf{CC}(G_i), E^C)$ is a forest, where $A, B \in \mathsf{CC}(G_i)$ are adjacent in $H^C$ if and only if one accepts the other or one is matched with the other. Moreover, each tree in $H^C$ has a diameter at most 3.*

**Proof.** First, we call the edges from the accepted request $E^C_a$ and the edges from the matching $E^C_m$. Then $E^C = E^C_a \cup E^C_m$. Since each cluster only initiates one request and we only accept requests from Blue to Red, $E^C_a$ induces a forest. Otherwise, there will be a cycle which consists of clusters of alternating colors. Then each cluster will be accepting some requests since each cluster only sends one request. However, this is impossible due to our acceptance rule because each accepting cluster (Red) will not be accepted, and each accepted cluster (Blue) will not be accepting any cluster. Since $E^C_m$ comes from matching the rejected nodes, and all rejected clusters are not connected by accepting edges, $H^C$ is a forest.

For the diameter, see that the connected components induced by $E^C_a$ are stars centered at a Red cluster. This is because each Blue cluster rejects all requests and each request from a Red cluster is rejected. Then, we perform a case analysis for the components connected by $E^C_m$. Let $\{A, B\} \in E^C_m$. If $A, B$ are both rejected Blue clusters, then this component has diameter 1. If $A, B$ are both rejected Red clusters, then this component has a diameter at most 3. If $A$ is Red and $B$ is Blue, this component has diameter at most 2. ◀

Define $H = (V, E(G_i) \cup E_a \cup E_m)$, where $E_a$ are edges from the *accepting messages* and $E_m$ are edges from the *regrouping messages*. We call each connected component in $H$ a grouped cluster in phase $i$. Note that each grouped cluster has agreed on a unique leader. We will now perform the merging step to transform each grouped cluster into a satisfactory tree.

To illustrate the grouping step clearly, we show a possible execution of the grouping step in Figure 1.

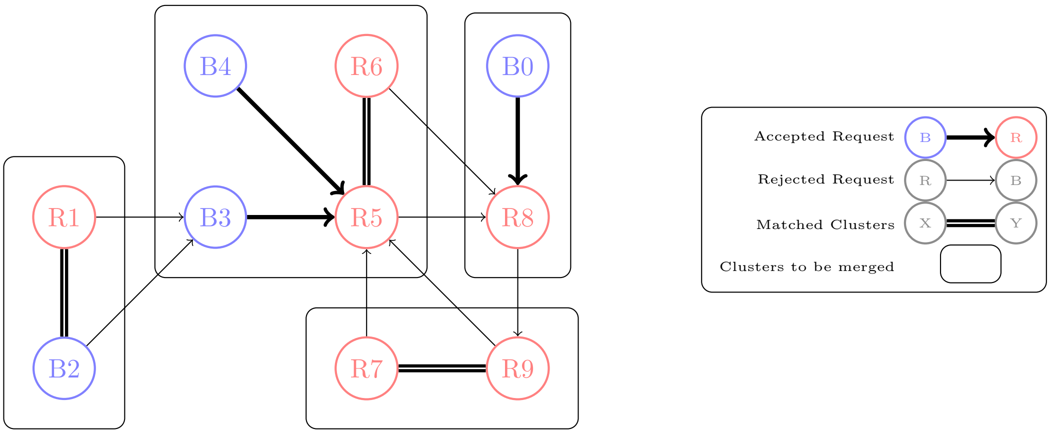

**Figure 1** A possible grouping step.

**Merging step.** Each node in a cluster that acts on behalf of the leader (to reply to requests) will inform its leader that it is the new leader of a grouped cluster. Each new leader will initiate a re-rooting process via a breadth-first search style broadcast, where each node will change its parent and children according to their distance from the new leader. Those requesting nodes that received an accepting message will re-root the requesting cluster towards the new leader by relaying this broadcast in the requesting cluster.

Let $v$ be the new leader of the grouped cluster. We now have a tree $T_v$ rooted at $v$ after re-rooting. $T_v$ has depth $O(\log n)$, since each original cluster has depth $O(\log n)$ and $H^C$ has diameter at most 3 due to Lemma 4.1. However, this tree might have a maximum degree up to $O(\Delta)$, due to the accepting edges i.e., edges in $E_a$. Therefore, we will perform the following transformation similar to the merging step in [24] to maintain the invariant. However, we made some crucial adaptation to the pointer jumping step to reduce message complexity at the cost of introducing randomness. First, we transform the tree into a child-sibling tree. Each node $v$ will sort its children in some arbitrary order and then attach itself at the head of this order. Then for each child $u$ in this order, $v$ will send the previous and next node in the order. The last node will receive no next node. In this way, each node keeps at most one child and one sibling. By viewing the sibling as a child, we have constructed a binary tree. Then, we can proceed with the same Euler Tour technique to turn this into a cycle of virtual nodes. Finally, we perform a procedure RC2T (described in the full version [14]) to turn this cycle of virtual nodes into a tree with $O(\log n)$ degrees and $O(\log n)$ depth.

The above steps to construct a satisfactory tree after re-rooting are described in more detail in the full version [14], where we show that running this process for $O(\log n)$ times can be done in $O(\log^2 n)$ rounds with $O(n \log n)$ messages. Moreover, each node $v$ uses at most $O(\deg_G(v) + \log n)$ messages.

**Post-processing.** After all the phases terminates, we get a cluster with a satisfactory tree overlay. We can now run one iteration of the deterministic well-formed tree construction in the full version to turn this satisfactory tree to a well-formed tree with additive $O(\log n)$ rounds and $O(n \log n)$ messages.

The formal analysis of the protocol HybridWFT is described in the full version [14], hence proving Theorem 1.4.

▶ **Theorem 1.4.** *There is a protocol in the* HYBRID $(\log n, 1, \log n)$ *model that can construct a well-formed tree overlay from any input graph $G$ in $O\left(\log^2 n\right)$ rounds with $O\left(n \log n\right)$ messages w.h.p. Moreover, each node $v$ sends and receives at most $O(\deg_G(v) + \log n)$ messages throughout the protocol.*

## References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 459–467. SIAM, 2012. doi:10.1137/1.9781611973099.40.

2 Emmanuelle Anceaume, Maria Gradinariu, and Aina Ravoaja. Incentives for p2p fair resource sharing. In *Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 253–260. IEEE, 2005. doi:10.1109/P2P.2005.17.

3 Dana Angluin, James Aspnes, Jiang Chen, Yinghua Wu, and Yitong Yin. Fast construction of overlay networks. In *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 145–154, 2005. doi:10.1145/1073970.1073991.

4 James Aspnes and Gauri Shah. Skip graphs. In *Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 384–393. SIAM, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644170.

5 John Augustine, Mohsen Ghaffari, Robert Gmyr, Kristian Hinnenthal, Christian Scheideler, Fabian Kuhn, and Jason Li. Distributed computation in node-capacitated networks. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, pages 69–79, 2019. doi:10.1145/3323165.3323195.

6 John Augustine, Kristian Hinnenthal, Fabian Kuhn, Christian Scheideler, and Philipp Schneider. Shortest paths in a hybrid network model. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1280–1299. SIAM, 2020. doi:10.1137/1.9781611975994.78.

7 John Augustine, Anisur Rahaman Molla, Ehab Morsy, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Storage and search in dynamic peer-to-peer networks. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 53–62, 2013. doi:10.1145/2486159.2486170.

8 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. doi:10.1145/2484239.2484275.

9 John Augustine, Gopal Pandurangan, and Peter Robinson. Fast byzantine leader election in dynamic networks. In *29th International Symposium on Distributed Computing (DISC)*, volume 9363 of *Lecture Notes in Computer Science*, pages 276–291, 2015. doi:10.1007/978-3-662-48653-5_19.

10 John Augustine, Gopal Pandurangan, Peter Robinson, Scott Roche, and Eli Upfal. Enabling robust and efficient distributed computation in dynamic peer-to-peer networks. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 350–369. IEEE, 2015. doi:10.1109/FOCS.2015.29.

11 John Augustine, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Towards robust and efficient computation in dynamic peer-to-peer networks. In *Proceedings of the Twenty-*

*third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 551–569, 2012. doi:10.1137/1.9781611973099.47.

12 John Augustine and Sumathi Sivasubramaniam. Spartan: A framework for sparse robust addressable networks. In *Proc. of the 32nd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1060–1069. IEEE, 2018. doi:10.1109/IPDPS.2018.00115.

13 Andrew Berns, Sukumar Ghosh, and Sriram V. Pemmaraju. Building self-stabilizing overlay networks with the transitive closure framework. *Theoretical Computer Science*, 512:2–14, 2013. doi:10.1016/J.TCS.2013.02.021.

14 Yi-Jun Chang, Yanyu Chen, and Gopinath Mishra. Overlay network construction: Improved overall and node-wise message complexity. *CoRR*, abs/2412.04771, 2024. doi:10.48550/arXiv.2412.04771.

15 Yi-Jun Chang, Oren Hecht, Dean Leitersdorf, and Philipp Schneider. Universally optimal information dissemination and shortest paths in the hybrid distributed model. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 380–390, 2024. doi:10.1145/3662158.3662791.

16 Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986. doi:10.1016/S0019-9958(86)80023-7.

17 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. URL: http://dl.acm.org/citation.cfm?id=1070432.1070574.

18 Maximilian Drees, Robert Gmyr, and Christian Scheideler. Churn- and dos-resistant overlay networks based on network reconfiguration. In *Proc. of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 417–427. ACM, 2016. doi:10.1145/2935764.2935783.

19 Fabien Dufoulon, Michael Moorman, William K Moses Jr, and Gopal Pandurangan. Time-and communication-efficient overlay network construction via gossip. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ITCS.2024.42.

20 Michael Feldmann, Christian Scheideler, and Stefan Schmid. Survey on algorithms for self-stabilizing overlay networks. *ACM Computing Surveys*, 53(4):1–34, 2020.

21 Hillel Gazit. An optimal randomized parallel algorithm for finding connected components in a graph. *SIAM Journal on Computing*, 20(6):1046–1067, 1991. doi:10.1137/0220066.

22 Seth Gilbert, Gopal Pandurangan, Peter Robinson, and Amitabh Trehan. Dconstructor: Efficient and robust network construction with polylogarithmic overhead. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 438–447. ACM, 2020. doi:10.1145/3382734.3405716.

23 C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks: Algorithms and evaluation. *Performance Evaluation*, 63(3):241–263, 2006.

24 Robert Gmyr, Kristian Hinnenthal, Christian Scheideler, and Christian Sohler. Distributed monitoring of network properties: The power of hybrid networks. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.137.

25 Thorsten Götte, Kristian Hinnenthal, and Christian Scheideler. Faster construction of overlay networks. In *International Colloquium on Structural Information and Communication Complexity*, pages 262–276. Springer, 2019. doi:10.1007/978-3-030-24922-9_18.

26 Thorsten Götte, Kristian Hinnenthal, Christian Scheideler, and Julian Werthmann. Time-optimal construction of overlay networks. *Distributed Computing*, 36(3):313–347, 2023. doi:10.1007/S00446-023-00442-4.

27 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

28 Riko Jacob, Andréa W. Richa, Christian Scheideler, Stefan Schmid, and Hanjo Täubig. Skip+: A self-stabilizing skip graph. *Journal of the ACM*, 61(6):36:1–36:26, 2014. doi:10.1145/2629695.

29 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. doi:10.1007/S12083-012-0124-Z.

30 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for $l_p$ samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 49–58, Athens, Greece, 2011. ACM. doi:10.1145/1989284.1989289.

31 Valerie King, Shay Kutten, and Mikkel Thorup. Construction and impromptu repair of an MST in a distributed network with $o(m)$ communication. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 71–80, 2015. doi:10.1145/2767386.2767405.

32 Fabian Kuhn and Philipp Schneider. Computing shortest paths and diameter in the hybrid network model. In *Proc. of the 39th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 109–118. ACM, 2020. doi:10.1145/3382734.3405719.

33 Vahid Heidaripour Lakhani, Leander Jehl, Rinke Hendriksen, and Vero Estrada-Galiñanes. Fair incentivization of bandwidth sharing in decentralized storage networks. In *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 39–44. IEEE, 2022. doi:10.1109/ICDCSW56584.2022.00017.

34 C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *IEEE INFOCOM*, pages 2133–2143, 2003.

35 Peter Mahlmann 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. doi:10.1145/1148109.1148162.

36 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. doi:10.1145/3382734.3405704.

37 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2$^{nd}$ edition, 2005. doi:10.1017/CBO9780511813603.

38 Jelani Nelson and Huacheng Yu. Optimal lower bounds for distributed and streaming spanning forest computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1844–1860. SIAM, 2019. doi:10.1137/1.9781611975482.111.

39 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. doi:10.1109/SFCS.2001.959925.

40 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. Fast distributed algorithms for connectivity and MST in large graphs. *ACM Transactions on Parallel Computing (TOPC)*, 5(1):1–22, 2018. doi:10.1145/3209689.

41 Peter Robinson. Being fast means being chatty: The local information cost of graph spanners. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2105–2120. SIAM, 2021. doi:10.1137/1.9781611976465.126.

42 Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350. Springer, 2001. doi:10.1007/3-540-45518-3_18.

43 Ion Stoica, Robert Tappan Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 149–160. ACM, 2001. doi:10.1145/383059.383071.

44 Huacheng Yu. Tight distributed sketching lower bound for connectivity. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1856–1873. SIAM, 2021. doi:10.1137/1.9781611976465.111.

## A Related works

Various studies have explored methods for transforming arbitrary connected graphs into specific target topologies, such as expanders and well-formed tree [3, 24, 25, 26, 19]. Angluin et al. [3] were among the first to address this problem, demonstrating that any connected graph $G$ with $n$ nodes and $m$ edges can be converted into a binary search tree with depth $O(\log n)$. Their algorithm requires $O(\Delta + \log n)$ rounds and $O(n(\Delta + \log n))$ messages, where $\Delta$ is the maximum degree of any node in the initial graph. The model they used allows each node to send only one message per round to a neighbor, and the resulting binary tree can be further transformed into other desirable structures like expanders, butterflies, or hypercubes. If the nodes are capable of sending and receiving a $O(\Delta \log n)$ number of messages per round, i.e. in P2P-CONGEST model, there exists a deterministic algorithm that operates in $O(\log^2 n)$ rounds, as shown in [24]. Recently, this has been improved to $O(\log^{3/2} n)$ rounds with high probability, as demonstrated in [25] for graphs with $\Delta$ as polylogarithmic.

Gilbert et al. [22] developed a different approach by designing a distributed protocol that efficiently reconfigures any connected network into a desired topology – such as an expander, hypercube, or Chord – with high probability. Here a node can send messages to all their neighbors in a round, regardless of their degree, resulting in faster communication for higher-degree nodes. Their protocol operates in $O(\text{polylog}\, n)$ rounds, utilizing messages of size $O(\log n)$ bits per link per round and achieving a message complexity of $\tilde{\Theta}(m)$.

Götte et al. [26] later introduced an algorithm for constructing a well-formed tree – a rooted tree with constant degree and $O(\log n)$ diameter – from any connected graph. Their protocol first builds an $O(\log n)$-degree expander, which can be further refined into the desired tree structure. The algorithm is optimal in terms of time, completing in $O(\log n)$ rounds, which aligns with the theoretical lower bound of $\Omega(\log n)$ for constructing such topologies from arbitrary graphs [26]. However, the message complexity remains $\tilde{\Theta}(m)$, as nodes are required to send and receive $d \log n$ messages per round, where $d$ is the initial maximum degree. The key innovation in their approach is the use of short random walks to systematically improve the conductance of the graph, ultimately leading to the formation of robust expander structures. Very recently, Dufoulon et al. [19] introduces GOSSIP-reply$(\log^2 n)$ model and showed that a constant-degree expander can be constructed starting from any initial topology by spending $O(\log^5 n)$ rounds and with message complexity $O(n \log^5 n)$. This algorithm in Dufoulon et al. is the first protocol that achieves message complexity independent of $m$. Note that our result on GOSSIP-reply model (i.e., Corollary 1.3) is a strict improvement over the result of Dufoulon et al. in terms of both round and message complexity.

The HYBRID$(\alpha, \beta, \gamma)$ model, initially introduced by [6] for studying shortest paths, was further explored by [26], who showed that in the HYBRID$(\log n, 1, \log^2 n)$ model, an arbitrary topology can be transformed into a well-formed tree within $O(\log n)$ rounds. The message complexity of their algorithm is $O(m + n \log^3 n)$. In contrast, our result in the HYBRID$(\log n, 1, \log n)$ model (Theorem 1.4) achieves communication efficiency, albeit in $O(\log^2 n)$ rounds.

Research on overlay construction extends well beyond simple foundational examples, mainly due to the inherently dynamic nature of real-world overlay networks, which are often impacted by churn and adversarial behaviors. This research can be categorized into two primary areas: self-stabilizing overlays and synchronous overlay construction algorithms. Self-stabilizing overlays, which locally detect and correct invalid configurations, are extensively surveyed by Feuilloley et al. [20]. However, many of these algorithms lack definitive communication complexity bounds and provide limited guarantees for achieving polylogarithmic rounds [13, 28]. On the other hand, synchronous overlay construction algorithms are designed to preserve the desired network topology despite the presence of randomized or adversarial disruptions, thereby ensuring efficient load balancing and generating unpredictable topologies under certain error conditions [10, 18, 12, 25]. A significant advancement in this area is made by Gilbert et al. [22], who demonstrated how fast overlay construction can be maintained even in the presence of adversarial churn, assuming the network stays connected and stable for an adequate duration. Additionally, Augustine et al. [3] investigated graph realization problems, focusing on rapidly constructing graphs with specific degree distributions; however, their approach assumes the initial network is arranged as a line, which simplifies the task. The complexity of overlay construction increases when nodes have restricted communication capabilities, prompting research into the Node-Capacitated Clique (NCC) model, where each node can send and receive $O(\log n)$ messages per round [5]. Within the NCC model, efficient algorithms have been developed for various local problems such as MIS, matching, coloring, BFS tree, and MST [5]. Notably, Robinson [41] established that constructing constant stretch spanners within this model necessitates polynomial time. Similar complexities are encountered in hybrid network models that blend global overlay communication with traditional frameworks like LOCAL and CONGEST, where extensive communication abilities enable solving complex problems like APSP and SSSP effectively, though often with considerable local communication overheads [6, 15, 32, 20].

## B Tradeoffs between complexity measures

There is a tradeoff among round complexity, message complexity, and balanced communication: If each node in $G$ is only allowed to send $O(1)$ messages per round, then any algorithm to build a constant-degree overlay $H$ requires $\Omega(\Delta(G))$ rounds [3]. To achieve both low round complexity and low message complexity independent of the initial degree of the input graph, certain nodes may need to send many more messages than others. This is observed in our first protocol in the GOSSIP-reply model in Section 3. Although it achieves good round and message complexities, it suffers from high regional communication where some nodes may need to communicate up to $\Omega(n)$ messages.

**Node-wise message complexity.** Node-wise message complexity can be seen as a measure that quantifies the aforementioned imbalance. The study of node-wise complexity is further motivated by the pursuit of fairness in the P2P network context. An extensive body of work is devoted to designing fair mechanisms to encourage users to contribute to the P2P network [2, 33]. However, besides offering incentives, it is essential to guarantee that users will incur low and fair costs when they join the network. A node may be discouraged from joining the network if it may potentially be selected as a crucial node of the network and perform much more work than other participating nodes.

**Communication capacity versus round complexity.** Communication capacity refers to the number of messages sent per node per round. This parameter captures the congestion in real-world networks. Intuitively, algorithms designed with more stringent communication capacity can perform better in real-world networks under congestion. There has been a series of works on improving the round complexity of the overlay network construction problem with more stringent communication capacity. Specifically, Angluin et al. [3] asked if there is an $O(\log n)$-round algorithm with $O(\Delta)$ communication capacity. While Götte et al. [26] answered this question affirmatively for the case that the communication capacity is $\Theta\left(\Delta + \log^2 n\right)$, in this work we present another tradeoff with $O\left(\log^2 n\right)$ round complexity and $O(\Delta + \log n)$ communication capacity, see Theorem 1.4 and Corollary 1.6.