OBST: A Self-Adjusting Peer-to-Peer Overlay Based on Multiple BSTs

Chen Avin[1], Michael Borokhovich[1], Stefan Schmid[2]
[1] Ben Gurion University, Beersheva, Israel; [2] TU Berlin & T-Labs, Berlin, Germany
{avin,borokhovich}@cse.bgu.ac.il; stefan@net.t-labs.tu-berlin.de

***Abstract*—The design of scalable and robust overlay topologies has been a main research subject since the very origins of peer-to-peer (p2p) computing. Today, the corresponding optimization tradeoffs are fairly well-understood, at least in the static case and from a worst-case perspective.**

This paper revisits the peer-to-peer topology design problem from a self-organization perspective. We initiate the study of topologies which are *optimized to serve the communication demand*, or even self-adjusting as demand changes. The appeal of this new paradigm lies in the opportunity to be able to go beyond the lower bounds and limitations imposed by a static, communication-oblivious, topology. For example, the goal of having short routing paths (in terms of hop count) does no longer conflict with the requirement of having low peer degrees.

We propose a simple overlay topology OBST(k) which is composed of k (rooted and directed) *Binary Search Trees (BSTs)*, where k is a parameter. We first prove some fundamental bounds on what can and cannot be achieved optimizing a topology towards a *static* communication pattern (a static OBST(k)). In particular, we show that the number of BSTs that constitute the overlay can have a large impact on the routing costs, and that a single additional BST may reduce the amortized communication costs from $\Omega(\log n)$ to $O(1)$, where n is the number of peers. Subsequently, we discuss a natural self-adjusting extension of OBST(k), in which frequently communicating partners are "splayed together".

I. Introduction

Classic literature on the design of peer-to-peer (p2p) topologies typically considers the optimization of *static* properties in the *worst case*, e.g., the maximal peer degree or the network diameter. An appealing alternative is to optimize the *amortized* performance of a p2p system (or more generally, a *distributed data structure*) based on the communication or usage patterns, either statically (based on known traffic statistics) or dynamically, exploiting temporal localities for self-adjustments.

One of the main metrics to evaluate the performance of a self-adjusting network is the amortized cost: the worst-case communication cost over time and per request. Splay trees are the most prominent example of the self-adjustment concept in the context of classic data structures: in their seminal work, Sleator and Tarjan [13] proposed self-adjusting binary search trees where popular items or *nodes* are moved closer to the *root* (where the lookups originate), exploiting potential non-uniformity in the access patterns.

Our Contributions. This paper initiates the study of how to extend the splay tree concepts [5], [13] to multiple trees, in order to design self-adjusting p2p *overlays*. Concretely, we propose a *distributed variant* of the splay tree to build the OBST overlay: in this overlay, frequently communicating partners are located (in the static case) or moved (in the dynamic case) topologically close(r), without sacrificing local routing benefits: While in a standard *binary search tree (BST)* a request always originates at the root (we will refer to this problem as the *lookup problem*), in the distributed BST variant, *any pair* of nodes in the network can communicate; we will refer to the distributed variant as the *routing problem*.

The reasons for focusing on BSTs are based on their simplicity and powerful properties: they naturally support local, greedy routing, they are easily self-adjusted, they support join-leave operations in a straight-forward manner, and they require low peer degrees. The main drawback is obviously the weak robustness imposed by the tree structure, and we address this by using multiple trees.

The proposed OBST(k) overlay consists of set of k *distributed BSTs*. (See Figure 1 for an example of a OBST(2).) We first study how the communication cost in a static OBST(k) depends on the number k of BSTs, and give an upper bound which shows that the overlay strictly improves with larger k. In fact, we will show that in some situations, changing from k to $k+1$ BSTs can make a critical difference in the *routing* cost. Interestingly, such a drastic effect is not possible on the classical *lookup* operations in a BST. This demonstrates that the problem of optimizing *routing* on a BST has some key differences from the *lookup* problem that was, and still is, extensively researched.

After studying the static case, we also describe a dynamic and self-adjusting variant of OBST which is inspired by classic splay trees: communication partners are topologically "splayed together". These splay operations are completely local and hence efficient.

II. Model and Definitions

We describe the p2p overlay network as a graph $\mathcal{H} = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is the set of peers and E represents their connections. For simplicity, we will refer by v_i both to the corresponding peer as well as the

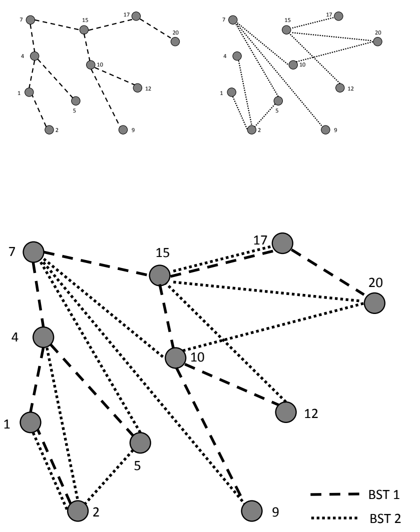

Fig. 1: Example of $\textsc{Obst}(2)$ consisting of two BSTs. Top left: BST 1 (e.g., rooted at peer v_7). Top right: BST 2 (e.g., rooted at peer v_{10}). Bottom: combined BSTs.

peer's (unique) identifier; sometimes, we will simply write i instead of v_i. Moreover, we will focus on bidirected overlays, i.e., we will ensure that if a peer $v_1 \in V$ is connected to another peer $v_2 \in V$, denoted by (v_1, v_2), then also v_2 is connected to v_1 (i.e., (v_2, v_1)). Sometimes we will refer to the two bidirected edges (v_1, v_2) and (v_2, v_1) simply by $\{v_1, v_2\}$.

We will assume that peers communicate according to a certain pattern. This pattern may be *static* in the sense that it follows a certain probability distribution; or it may be *dynamic* and change arbitrarily over time. Static communication patterns may conveniently be represented as a weighted directed graph $\mathcal{G} = (V, E)$: any peer pair (v_1, v_2) communicating with a non-zero probability is connected in the graph $\mathcal{G}$.

We will sometimes refer to the sequence of communication events between peers as *communication requests* σ. In the static case, we want the overlay $\mathcal{H}$ be as similar as possible to the communication pattern $\mathcal{G}$ (implied by σ), in the sense that an edge $e \in E(\mathcal{G})$ is represented by a short route in $\mathcal{H}$; this can be seen as a graph embedding problem of $\mathcal{G}$ (the "guest graph") into $\mathcal{H}$ (the "host graph"). In the dynamic setting, the topology $\mathcal{H}$ can be adapted over time depending on σ. These topological transformations should be *local*, in the sense that only a few peers and links in a small subgraph are affected.

Our proposed topology $\textsc{Obst}(k)$ can be described by a simple graph $\mathcal{H}$ which consists of a set of k *binary search trees (BST)*, for some $k > 0$.

Definition 1 ($\textsc{Obst}(k)$)**.** *Consider a set* $\{T_1, \ldots, T_k\}$ *of* k *BSTs.* $\textsc{Obst}(k)$ *is an overlay over the peer set* $V = \{1, \ldots, n\}$ *where connections are given by the BST edges, i.e.,* $E = \bigcup_{i=1}^{k} E(T_i)$.

Our topological transformations to adapt the $\textsc{Obst}(k)$ are *rotations* over individual BSTs: minimal and local transformations that preserve a BST. Informally, a rotation in a sorted binary search tree changes the local order of three connected nodes, while keeping subtrees intact. Note that it is possible to transform any binary search tree into any other binary search tree by a sequence of local transformations (e.g., by induction over the subtree roots).

Let $\sigma = (\sigma_0, \sigma_1 \ldots)$ be a sequence of m *requests*. Each request $\sigma_t = (u, v)$ is a pair of a source peer and a destination peer. Let $\mathcal{A}$ be an algorithm that given the request σ_t and the graph $\mathcal{H}_t$ at time t, transforms the current graph (via local transformations) to $\mathcal{H}_{t+1}$ at time $t + 1$. We will use $\textsc{Stat}$ to refer to an any static (i.e., non-adjusting) "algorithm" which does not change the communication network over time; however, $\textsc{Stat}$ is initially allowed to choose an overlay which reflects the statistical communication pattern.

The cost of the network transformations at time t are denoted by $\rho(\mathcal{A}, \mathcal{H}_t, \sigma_t)$ and capture the number of rotations performed to change $\mathcal{H}_t$ to $\mathcal{H}_{t+1}$; when $\mathcal{A}$ is clear from the context, we will simply write ρ_t. We denote with $\mathrm{d}_{\mathcal{H}}(\cdot)$ the distance function between nodes in $\mathcal{H}$, i.e., for two nodes $v, u \in V$ we define $\mathrm{d}_{\mathcal{H}}(u, v)$ to be the number of edges of a *shortest* path between u and v in $\mathcal{H}$. (The subscript $\mathcal{H}$ is optional if clear from the context.) Note that for a BST T, the shortest path between u and v is unique and can be found and routed locally via a greedy algorithms.

For a given sequence of communication requests, the cost for an algorithm is given by the number of transformations and the distance of the communication requests. Formally, we will make use of the following standard definitions (see also [5]).

Definition 2 (***Average and Amortized Cost***)**.** *For an algorithm* $\mathcal{A}$ *and given an initial network* $\mathcal{H}_0$ *with node distance function* $\mathrm{d}(\cdot)$ *and a sequence* $\sigma = (\sigma_0, \sigma_1 \ldots \sigma_{m-1})$ *of communication requests over time, we define the* (average) cost *of* $\mathcal{A}$ *as:* $\mathrm{Cost}(\mathcal{A}, \mathcal{H}_0, \sigma) = \frac{1}{m} \sum_{t=0}^{m} (\mathrm{d}_{\mathcal{H}_t}(\sigma_t) + 1 + \rho_t)$ *The* amortized cost *of* $\mathcal{A}$ *is defined as the worst possible cost of* $\mathcal{A}$*, i.e.,* $\max_{\mathcal{H}_0, \sigma} \mathrm{Cost}(\mathcal{A}, \mathcal{H}_0, \sigma)$.

One may consider two different routing models on $\textsc{Obst}$. In the first model, two peers will always communicate along a *single* BST: one which minimizes the hop length; the best BST may be found, e.g., via a probe message along the trees: the first response is taken. In the second model, we allow routes to cross different BSTs, and take the globally shortest path; this can be achieved, e.g., by using a standard routing protocol (e.g., distance vector) in the background. In the following, if not stated differently, we will focus on the first model, which is more conservative in the sense that it yields higher costs.

III. Background on BSTs and Splay Trees

The following facts are useful in the remainder of this paper. Theorem 1 bounds the lookup cost in an optimal binary search tree under a given *lookup* sequence σ: a sequence of requests all originating from the root of the tree.

Theorem 1 ([12])**.** *Given σ, for any (optimal) BST T, the amortized cost is at least*

$$\text{Cost}(\text{STAT}, T, \sigma) \geq \frac{1}{\log 3} H(\hat{Y}) \tag{1}$$

where $\hat{Y}(\sigma)$ is the empirical measure of the frequency distribution of σ and $H(\hat{Y})$ is its empirical entropy.

Knuth [10] fist gave an algorithm to find an optimal BST, and Mehlhorn [12] proved that a simple greedy algorithm is near optimal with an explicit bound:

Theorem 2 ([12])**.** *Given σ, there is a BST, T_{bal} that can be computed using a balancing argument and has an amortized cost of at most*

$$\text{Cost}(\text{STAT}, T_{\text{bal}}, \sigma) \leq 2 + \frac{H(\hat{Y})}{1 - \log(\sqrt{5} - 1)} \tag{2}$$

where $\hat{Y}(\sigma)$ is the empirical measure of the frequency distribution of σ and $H(\hat{Y})$ is its empirical entropy.

Sleator and Tarjan were able to show that splay trees, a self-adjusting BST based on an algorithm ST, yields the same amortized cost as an optimal binary search tree.

Theorem 3 (Static Optimality Theorem [13] - rephrased)**.** *Let σ be a sequence of lookup requests where each item is requested at least once, then for any initial tree T $\text{Cost}(\text{ST}, T, \sigma) = O(H(\hat{Y}))$ where $H(\hat{Y})$ is the empirical entropy of σ.*

In [5], Avin et al. proposed a *single* dynamic splay BST for routing, and a *double splay algorithm* DS. For the single tree case and any initial tree T the authors proved the following lower bound for a static solution STAT:

$$\textit{Cost}(\text{STAT}, T, \sigma) = \Omega(H(\hat{Y}|\hat{X}) + H(\hat{X}|\hat{Y}))$$

and the following upper bound for DS:

$$\textit{Cost}(\text{DS}, T, \sigma) = O(H(\hat{X}) + H(\hat{Y}))$$

where $\hat{X}$ and $\hat{Y}$ are the empirical measures of the frequency distribution of the sources and destinations from σ, respectively and H is the entropy function.

It is easy to see that BSTs support simple and local routing. [5]

Claim 1. *BSTs support local routing.*

IV. Static Obst(k) Optimization for P2P

We will first study static overlay networks which are optimized towards a request distribution given beforehand. The number of BSTs k is given together with the sequence of communication requests $\sigma = (\sigma_0, \sigma_1, \ldots)$. The goal is to find the optimal $\text{OBST}(k)$ to minimize *Cost*$(\text{STAT}, \text{OBST}(k), \sigma)$.

In [5] it is was proved that for any σ, the optimal $\text{OBST}(1)$ can be found in polynomial time. Here we first provide a new upper bound for the optimal $\text{OBST}(k)$ and show how it can improve with k.

For communication requests σ let $x_i(\sigma)$ (or for short x_i) be the frequency of v_i being a *source* in σ; similarly let y_i be the frequency of v_i being a *destination*, and let f_{ij} denote the frequency of the request (v_i, v_j) in σ. Define $z_i = (x_i + y_i)/2$ and note that by definition $\sum_1^n z_i = 1$. Let $\hat{Z}$ be a random variable (r.v.) with a probability distribution defined by the $z_i's$. For any k-partition of the requests in σ into disjoint sets $S_1, S_2, \ldots, S_k$, let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be the frequency measure of the partition, i.e., $\alpha_i = \sum_{(i,j)\in S_i} f_{ij}$.

First we can prove a new bound on the optimal static $\text{OBST}(1)$:

Theorem 4. *Given σ, there exists a $\text{OBST}(1)$ such that:*

$$\text{Cost}(\text{STAT}, \text{OBST}(1), \sigma) \leq 4 + \frac{2H(\hat{Z})}{1 - \log(\sqrt{5} - 1)}$$

where $H(\hat{Z})$ is the entropy of $\hat{Z}$ as defined earlier.

Consider now the $\text{OBST}(k)$ overlay which consists of k BSTs. Assume again a non-optimal strategy: we partition σ into k disjoint sets of requests $S_1, S_2, \ldots, S_k$, and each request is routed on its unique BST. In each tree we use the previous method, and the messages are routed from the source to the root and from the root to the destination.

We can now prove an upper bound on $\text{OBST}(k)$ that improves with k.

Theorem 5. *Given σ, there exists a $\text{OBST}(k)$ such that:*

$$\text{Cost}(\text{STAT}, \text{OBST}(k), \sigma) \leq 4 + \frac{2H(\hat{Z}) - 2H(\alpha_1, \alpha_2, \ldots, \alpha_k)}{1 - \log(\sqrt{5} - 1)}$$

where $H(\hat{Z})$ is the entropy of $\hat{Z}$ as defined earlier.

Note that this approach can yield a cost reduction of up to $\log k$, when the α_i values are equal. The problem of equally partition σ into k sets in order to maximize $H(\alpha_1, \alpha_2, \ldots, \alpha_k)$ is NP-complete, since even the partition problem (i.e., $k = 2$) and in particular the balanced partition problem (with $k = 2$) are NP-complete [9]. Interestingly, for those cases, $k = 2$, a pseudo-polynomial time dynamic programming algorithm exists.

The bound in Theorem 5 is conservative in the sense that sometimes, a single additional BST can reduce the optimal communication cost of $\text{OBST}(k)$ from worst possible (e.g., $\Omega(\log n)$) to a constant cost in $\text{OBST}(k+1)$.

Theorem 6. *A single additional BST can reduce the amortized costs from a best possible value of $\Omega(\log n)$ to $O(1)$.*

Essentially, it follows from the two BSTs $T_1 = (V, E_1)$ and $T_2 = (V, E_2)$ shown in Figure 2: obviously, the two BSTs can be perfectly embedded into $\text{OBST}(2)$ consisting of two BSTs as well. However, embedding the two trees at low cost in one BST is impossible, since there is a large cut in the identifier space.

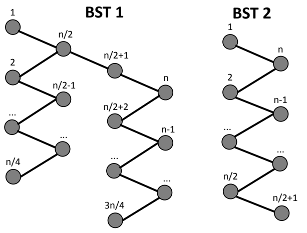

Fig. 2: A request sequence σ originating from these specific trees can yield high amortized costs.

Interestingly, such a high benefit from one additional BST is unique to the routing model and does not exist for classic lookup data structures as demonstrated by the following theorem.

Consider a sequence $\sigma = (v_0, v_1, \ldots, v_{m-1})$, $v_i \in V$ of *lookup* requests, and $|V| = n$. Theorem 1 can be generalized to k parallel lookup BSTs.

Theorem 7. *Given σ, for any* $\text{OBST}(k)$*:*

$$\text{Cost}(\text{STAT}, \text{OBST}(k), \sigma) \geq \frac{H(\hat{Y}) - \log k}{\log 3},$$

where $\hat{Y}(\sigma)$ is the empirical frequency distribution of σ.

V. Dynamic Self-Adjusting Obst(k) Overlay

Given our first insights on the performance of static $\text{OBST}(k)$ networks, let us now initiate the discussion of self-adjusting variants: BSTs which adapt to the demand, i.e., the sequence σ.

We initialize $\text{OBST}(k)$ as follows: each BST connects *all* peers V as a random and independent binary search tree. When communication requests occur, BSTs start to adapt. In the following, we will adjust the overlay at each interaction ("communication event" or "request") of two peers. Of course, in practice, such frequent changes are undesirable. While our protocol can easily be adapted such that peers only initiate the topological rearrangements after a certain number of interactions (within a certain time period), in order to keep our model simple, we do not consider these extensions here.

We propose a straight-forward *splay* method (inspired from the classical splay trees) to change the $\text{OBST}(k)$: whenever a peer u communicates with a peer v, we perform a distributed splay operation in *one* of the BSTs, namely in the BST T in which the two communication partners (u, v) are the topologically closest.

Concretely, upon a communication request (u, v), we determine the BST T (in case multiple trees yield similar cost, an arbitrary one is taken), as well as the least common ancestor w of u and v in T: $w := \text{LCA}_T(u, v)$. Subsequently, u and v are splayed to the root of the subtree (henceforth denoted by $T(w)$) of T rooted at w (a so-called *double-splay* operation [5]).

Algorithm 1 Dynamic $\text{OBST}(k)$

```
1: (* upon request $(u, v)$ *)
2: find BST $T \in \text{OBST}$ where $u$ and $v$ are closest;
3: $w := \text{LCA}_T(u, v)$;
4: $T' :=$ splay $u$ to root of $T(w)$;
5: splay $v$ to the child of $T'(u)$;
```

Figure 3 gives an example: upon a communication request between peers v_5 and v_{12}, the two peers are splayed to their least common ancestor, peer v_7, in BST T_1.

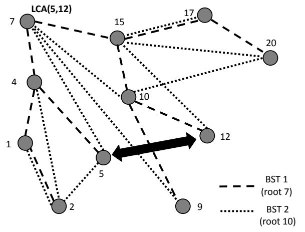

Fig. 3: Example for splay operation in $\text{OBST}(2)$ of Fig. 1.

Simulations: We conducted simulations to investigate the behavior of a self-adjusting $\text{OBST}(k)$. In Fig. 4 we can see how additional trees reduce the routing cost. All the requests were generated by a random perfect matching on various guest graphs: FB – Facebook (obtained from [14]), $\text{RND}(16)$ – random $\text{OBST}(16)$, and $\text{BAD}(2)$ – special worst case $\text{OBST}(2)$ as illustrated in Fig. 2. While we can see a steady improvement in the FB guest graph, for the $\text{RND}(16)$ guest graph the self-adjusting $\text{OBST}(16)$ achieves perfect convergence. In Fig. 4 (c) we can see a convergence of self-adjusting $\text{OBST}(2)$ to the $\text{BAD}(2)$, and an illustration of the case where one additional BST can make a significant difference in routing cost (similar to the static case of Theorem 6).

VI. Related Work

We are only aware of two papers on demand-optimized or self-adjusting overlay networks: Leitao et al. [11] study an overlay supporting gossip or epidemics on a dynamic topology. In contrast to our work, their focus is on unstructured networks (e.g., lookup or routing is not supported), and there is no formal evaluation. The paper closest to ours is [5]. Avin et al. initiate the study

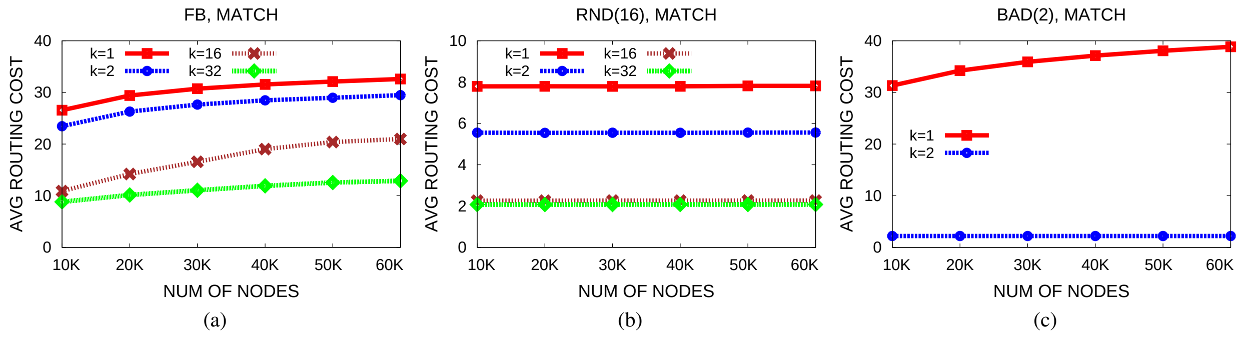

Fig. 4: Average routing distance in OBST(k) as a function of the number of BSTs k and the network size n.

of self-adjusting splay BSTs and introduce the double-splay algorithm. Although their work regards a distributed scenario, it focuses on a single BST only. Our work builds upon these results and investigates the benefits of having multiple trees, which is also more realistic in the context of p2p computing.

More generally, one may also regard geography [8] or latency-aware [7] p2p systems as providing a certain degree of self-adaptiveness. However, these systems are typically optimized towards more static criteria, and change less frequently. This also holds for the p2p topologies tailored towards the ISPs' infrastructures [2].

Our work builds upon classic data structure literature, and in particular on the splay tree concept [13]. Splay trees are optimized BSTs which move more popular items closer to the root in order to reduce the average access time. Regarding the splay trees as a network, [13] describes self-adjusting networks for *lookup* sequences, i.e., where the source is a *single (virtual) node* that is connected to the root. Splay trees have been studied intensively for many years (e.g. [3], [13]), and the famous dynamic optimality conjecture continues to puzzle researchers [6]: The conjecture claims that splay trees perform as well as any other binary search tree algorithm. Recently, the concurrent splay tree variant CBTrees [1] has been proposed. Unlike splay trees, CBTrees perform rotations infrequently and closer to the leaves; this improves scalability in multicore settings.

VII. Conclusion

This paper initiated the study of p2p overlays which are statically optimized for or adapt to specific communication patterns. We understand our algorithms and bounds as a first step, and believe that they open interesting directions for future research. For example, it would be interesting to study the multi-splay overlay from the perspective of online algorithms: While computing the competitive ratio achieved by classic splay trees (for lookup) arguably constitutes one of the most exciting open questions in Theoretical Computer Science [6], our work shows that the routing variant of the problem is rather different in nature (e.g., results in much lower cost). Another interesting research direction regards alternative overlay topologies: while we have focused on a natural BST approach, other graph classes such as the frequently used hypercubic networks and skip graphs [4] may also be made self-adjusting. Since these topologies also include tree-like subgraphs, we believe that our results may serve as a basis for these extensions accordingly.

References

[1] Y. Afek, H. Kaplan, B. Korenfeld, A. Morrison, and R. E. Tarjan. Cbtree: a practical concurrent self-adjusting search tree. In *Proc. 26th International Conference on Distributed Computing (DISC)*, pages 1–15, 2012.
[2] V. Aggarwal, A. Feldmann, and C. Scheideler. Can isps and p2p users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 37(3):29–40, 2007.
[3] B. Allen and I. Munro. Self-organizing binary search trees. *J. ACM*, 25:526–535, 1978.
[4] J. Aspnes and G. Shah. Skip graphs. *ACM Transactions on Algorithms (TALG)*, 3(4), 2007.
[5] C. Avin, B. Haeupler, Z. Lotker, C. Scheideler, and S. Schmid. Locally self-adjusting tree networks. In *Proc. 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2013.
[6] E. Demaine, D. Harmon, J. Iacono, and M. Patrascu. Dynamic optimality–almost. In *Proc. Annual Symposium on Foundations of Computer Science (FOCS)*, volume 45, pages 484–490, 2004.
[7] P. Druschel and A. Rowstron. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.
[8] C. Gross, D. Stingl, B. Richerzhagen, A. Hemel, R. Steinmetz, and D. Hausheer. Geodemlia: A robust peer-to-peer overlay supporting location-based search. In *Proc. 12th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 25–36, 2012.
[9] M. T. C. S. JIS. Computers and intractability a guide to the theory of np-completeness. 1979.
[10] D. Knuth. Optimum binary search trees. *Acta informatica*, 1(1):14–25, 1971.
[11] J. Leitao, J. Marques, J. Pereira, and L. Rodrigues. X-bot: A protocol for resilient optimization of unstructured overlay networks. *IEEE Transactions on Parallel and Distributed Systems*, 99, 2012.
[12] K. Mehlhorn. Nearly optimal binary search trees. *Acta Informatica*, 5(4):287–295, 1975.
[13] D. Sleator and R. Tarjan. Self-adjusting binary search trees. *Journal of the ACM (JACM)*, 32(3):652–686, 1985.
[14] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, pages 37–42, New York, NY, USA, 2009. ACM.