Brief Announcement: SplayNets

Towards Self-Adjusting Distributed Data Structures

Stefan Schmid[1], Chen Avin[2], Christian Scheideler[3],
Bernhard Haeupler[4], and Zvi Lotker[2]

[1] TU Berlin & T-Labs, Germany; [2] BGU, Israel; [3] U. Paderborn, Germany; [4] MIT, USA

Abstract. This paper initiates the study of self-adjusting distributed data structures or networks. In particular, we present *SplayNets*: a binary search tree based network that is self-adjusting to the routing requests. We derive entropy bounds on the amortized routing cost and show that our splaying algorithm has some interesting properties.

1. Distributed Splay Trees. In the mid 80s, Sleator and Tarjan [1] introduced an appealing new paradigm to design efficient data structures: rather than optimizing traditional metrics such as the search tree depth in the *worst-case*, the authors proposed to make data structures *self-adjusting* and considered the *amortized cost* as the performance metric—the "average cost" per operation for a given sequence s of lookups. The authors described *splay trees*, self-adjusting binary search trees in which frequently accessed elements are moved closer to the root, improving the average access times *weighted by the elements' popularity*. The popularity distribution must not be known in advance and may even change over time. We, in this paper, initiate the study of a *distributed generalization* of splay trees as a network. We consider a distributed data structure, e.g., a structured peer-to-peer (p2p) system or Distributed Hash Table (DHT), where nodes (i.e., "peers") that communicate more frequently should become topologically closer to each other (i.e., reducing the routing distance). This contrasts with most of today's structured peer-to-peer overlays whose topology is often optimized in terms of static global properties only, such as the node degree or the longest routing path.

2. Model and Problem Definition. Given an arbitrary and unknown pattern of communication (or routing) requests σ between a set of nodes $V = \{1, \ldots, n\}$, we attend to the problem of finding good *communication networks* G out of a family of *allowed networks* $\mathcal{G}$. Each topology $G \in \mathcal{G}$ is a graph $G = (V, E)$, and we define a set of *local transformations* on graphs in $\mathcal{G}$ to transform one member $G' \in \mathcal{G}$ to another member $G'' \in \mathcal{G}$. We seek to adapt our topologies smoothly over time, i.e., a changing communication pattern leads to "local" changes of the communication graph over time. We focus on the special case where $\mathcal{G}$ is the set of *binary search trees* (BST), henceforth simply called *BST networks*. Besides their simplicity, such networks are attractive for their low node degree and the possibility to route locally: given an destination identifier (or address), each node can decide locally whether to forward the packet to its left child, its right child, or its parent. The local transformations of BST networks are called *rotations*. Let $\sigma = (\sigma_0, \sigma_1 \ldots \sigma_{m-1})$ be a sequence of m *communication requests* where $\sigma_t = (u, v) \in V \times V$ denotes that a packet needs to be sent from a *source* u to a *destination* v. The cost of the network transformations at time t is denoted by $\rho(\mathcal{A}, G_t, \sigma_t)$ (or simply ρ_t) and captures the number of rotations performed to

M.K. Aguilera (Ed.): DISC 2012, LNCS 7611, pp. 439–440, 2012.

change G_t to G_{t+1}. We denote with $\mathrm{d}_G(\cdot)$ the distance function between nodes in G, i.e., for two nodes $v, u \in V$ we define $\mathrm{d}_G(u, v)$ to be the number of edges of a *shortest* path between u and v in G. 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 plus one. For an algorithm $\mathcal{A}$ and given an initial network G_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: $\mathit{Cost}(\mathcal{A}, G_0, \sigma) = \frac{1}{m}\sum_{t=0}^{m-1} (\mathrm{d}_{G_t}(\sigma_t) + 1 + \rho_t)$. The *amortized cost* of $\mathcal{A}$ is defined as the worst possible cost of $\mathcal{A}$, i.e., $\max_{G_0,\sigma} \mathit{Cost}(\mathcal{A}, G_0, \sigma)$.

3. SplayNets: Algorithm and Analysis. The main idea of our *double splay* algorithm DS is to perform splay tree operations in *subtrees* covering the different communication partners. Concretely, consider a communication request (u, v) from node u to node v, and let $\alpha_T(u, v)$ denote the least common ancestor of u and v in the current BST network T. Furthermore, for an arbitrary node x, let $T(x)$ denote the subtree rooted at x. The formal listing of DS is shown in Algorithm 1: When a request (u, v) occurs, DS first simply splays u to the least common ancestor $\alpha_T(u, v)$ of u and v, using the classic splay operations Zig, ZigZig, ZigZag from [1]. Subsequently, the idea is to splay the destination node v to the child of the least common ancestor $\alpha_{T'}(u, v)$ of u and v in the resulting tree T'. The communication cost of DS can be upper bounded by the empirical entropy of the sources and destinations of the requests. We can also provide a lower bound for any BST network based on the conditional empirical entropy.

Algorithm 1 Double Splay Algorithm DS

1: (* upon request (u, v) in T *)
2: $w := \alpha_T(u, v)$
3: $T' :=$ **splay** u to root of $T(w)$
4: **splay** v to the child of $T'(u)$

Fig. 1. Double Splay Algorithm DS

Theorem 1. *Let σ be an arbitrary sequence of communication requests, then for any initial BST T_0,* $\mathrm{Cost}(\mathtt{DS}, T_0, \sigma) \in O(H(\hat{X}) + H(\hat{Y}))$ *where $H(\hat{X})$ and $H(\hat{Y})$ are the empirical entropies of the sources and the destinations in σ, respectively.*

Theorem 2. *Given a request sequence σ, for any optimal BST network T:* $\mathrm{Cost}(\perp, T, \sigma) \in \Omega(H(\hat{Y}|\hat{X}) + H(\hat{X}|\hat{Y}))$.

A simple corollary of the above results can be obtained when σ follows a *product distribution* (i.e., $H(\hat{X}|\hat{Y}) = H(\hat{X})$): DS is asymptotically optimal if σ follows a product distribution. In our full paper, we will show that DS features several other desirable properties and that it is optimal in some special cases like when σ forms a *laminated set* or a *BST*. In addition we extend Theorem 2 to give more sophisticated lower bounds.

4. Discussion. We regard our work as a first step towards the design of novel distributed data structures and networks which adapt dynamically to the demand.

Reference

1. Sleator, D., Tarjan, R.: Self-adjusting binary search trees. JACM 32(3), 652–686 (1985)