The Name of the Title is Hope

BEN TROVATO* and G.K.M. TOBIN*, Institute for Clarity in Documentation, USA
LARS THØRVÄLD, The Thørväld Group, Iceland
VALERIE BÉRANGER, Inria Paris-Rocquencourt, France
APARNA PATEL, Rajiv Gandhi University, India
HUIFEN CHAN, Tsinghua University, China
CHARLES PALMER, Palmer Research Laboratories, USA
JOHN SMITH, The Thørväld Group, Iceland
JULIUS P. KUMQUAT, The Kumquat Consortium, USA

CCS Concepts: • **Theory of computation → Distributed algorithms**.

Additional Key Words and Phrases: Stretch, Spanning Tree, Overlay, Closeness centrality

ACM Reference Format:
Ben Trovato, G.K.M. Tobin, Lars Thørväld, Valerie Béranger, Aparna Patel, Huifen Chan, Charles Palmer, John Smith, and Julius P. Kumquat. 2018. The Name of the Title is Hope. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION AND RELATED WORK

An *overlay* is considered to be an abstraction of the existing underlying network (referred to as the *underlay*). The underlay is the backbone network and is usually static and non-scalable, whereas the overlay supports designers to have a level of dynamism and scalability in the network.

More formally, an overlay can be defined as a graph on a subset of the vertices of the underlay, and an overlay edge signifying a path in the underlying graph. From this definition, it is imperative that if (u, v) is an edge present in the overlay, then vertices u and v must be connected in the underlay.

In a usual sense, the network topology realized through the overlay, has some desirous properties: low diameter, low congestion and low mean path length for routing. It is expected that despite the graph being dynamic in nature, the "desirable" properties of the topology, i.e., the overlay, remain intact. Hence, graceful performance with respect to the construction and maintenance of the overlay, in an error-prone environment is quite a challenge.

It is noteworthy to again emphasize the fact that the overlay is a logical connection of nodes in the underlying topology, and the application for which the overlay is needed determines the design of the overlay. For example, the designer may choose to create a spanning tree as the overlay if he wants to maintain connectivity with minimal edges, or

*Both authors contributed equally to this research.

Manuscript submitted to ACM

he may want to have complete graph if minimization of communication latency is the priority. The network parameter that the designer wants to optimize, determines the choice of topology of the overlay.

The existing overlay construction algorithms focus on building network topologies such as skip-graphs, skip-lists, line graphs and rings [6, 8, 9] to name a few. They however aim to minimize only the diameter and node degrees in the final overlay, without being concerned with the underlay path length being incurred in such constructions. Having a low underlay latency is extremely important in various routing applications. Here, the concept of *stretch* comes in. Stretch is a parameter that helps us assess how much the shortest distance between a pair of nodes in the overlay is off from the minimum distance between them in the underlay. We aim to reduce this value of distortion. In this paper, we focus on building a low stretch spanning tree as the overlay.

It is noteworthy to again emphasize the fact that the overlay is a logical connection of nodes in the underlying topology, and the application for which the overlay is needed determines the design of the overlay. For example, the designer may choose to create a spanning tree as the overlay if he wants to maintain connectivity with minimal edges, or he may want to have complete graph if minimization of communication latency is the priority. The network parameter that the designer wants to optimize, determines the choice of topology of the overlay.

The existing well-known overlay construction algorithms focus on building network topologies such as skip-graphs, skip-lists, line graphs and rings [6, 8, 9] to name a few. They however aim to minimize only the diameter and node degrees in the final overlay, without being concerned with the underlay path length being incurred in such constructions. Having a low underlay latency is extremely important in various routing applications. Stretch is a parameter that helps us assess how much the shortest distance between a pair of nodes in the overlay is off from the minimum distance between them in the underlay. We aim to reduce this value of distortion. In this paper, we focus on building a low stretch spanning tree as the overlay.

The Minimum Max-Stretch Spanning Tree (MMST) [3] problem focuses on the parameter stretch mentioned before. We define stretch(σ) between two nodes u and v as:

$$\sigma(u,v) = \frac{dist_T(u,v)}{dist_{\hat{G}(u,v)}} \tag{1}$$

Here, $dist_T(u,v)$ is the distance between u and v in T. Whereas, $dist_{\hat{G}}(u,v)$ is the shortest distance between x and y in $\hat{G}$.

The parameter *max stretch* is defined as:

$$max - str(G) = min\{\max_{x,y \in V(G)} \{\sigma(x,y)\} \mid T \text{ is a spanning tree of } G\}. \tag{2}$$

The problem of constructing T that aims to minimize the parameter *max-str*(G) is precisely the Minimum Max-Stretch Spanning Tree (MMST) problem. It is known to be NP-Hard [2].

We claim the following significant **contributions** in this paper:

- We provide a novel self-stabilizing algorithm for the MAST problem. To the best of our knowledge, this is the only work that presents an approach to construct a low average stretch tree resilient to transient failures using non-masking fault-tolerance techniques like self-stabilization.
- The construction technique is loosely based on the low diameter decomposition on which the non-self-stabilizing version of [5] works.
- The proposed *BuildTree* protocol terminates in $O(n + \Delta \cdot \eta)$ rounds and correctly generates a low average stretch tree. We show that the average stretch of $n^{o(1)}$ as analyzed for the non-self-stabilizing algorithm in [5] also holds in our case.

The rest of the paper is organized as follows; preliminaries illustrates the system model and various notions. The distributed cluster tree used to design the proposed protocol is discussed in dct. const presents low average stretch spanning tree construction using our *BuildTree* algorithm. We finally, conclude the paper in conclusion.

2 PRELIMINARIES

https://cs.stackexchange.com/questions/119854/algorithm-for-finding-mean-center-of-unweighted-graph

We consider an undirected and unweighted graph $\hat{G}(V, E)$ as the underlay, where V is the set of nodes, and E is the set of edges. Further, let $|V| = n$ and $|E| = m$. The distance between two nodes x and y is the number of edges on the shortest path from x to y, and is denoted as $dist(x, y)$. A subscript is often used in conjunction with this variable in this paper, which simply depicts the network on which we refer this distance. In order to analyze nodes based on their relative position in the graph, we use the metric of *Farness* [4]. This parameter helps us determine the position of a node in comparison to its distance from the other nodes in the network. The Farness of a node is defined as:

$$Farness(u) = \sum_{\forall v \in V} dist(u, v)$$

Another centrality metric closely related to Farness is *Closeness*. We have the following definition from [4]:

$$Closeness(u) = \frac{1}{\sum_{\forall v \in V} dist(u, v)}$$

In other words, Closeness is precisely the reciprocal of Farness. Intuitively, greater the value of Closeness for a node, the closer it is to all the other nodes in the network. Hence, a node with low Farness and high Closeness is more "central" in a network.

An overlay $G(V, E')$ is defined with respect to the underlay graph $\hat{G}$. Note that E' is allowed to have all the possible $\binom{n}{2}$ edges corresponding to $\hat{G}$. However, we can connect nodes u and v in the overlay G via an edge $(u, v) \in E'$ only if there exists a path between u and v in the underlay $\hat{G}$. It is intuitive to understand that if the underlay $\hat{G}$ is connected, any edge can be added to the overlay without worrying about its connectedness criteria.

Let $T(V, E_T)$ be a spanning tree defined on $\hat{G}$. We note that the distance between two nodes x and y to be the hop count of the shortest path from x to y. This is denoted as $dist(x, y)$. The diameter ($Diam$) of the graph is denoted as: $Diam = \max_{u,v \in V} dist(u, v)$. Let $Neigh(u) = \{x | (u, x) \in E\}$ be the set of neighbours of node u in $\hat{G}$. Further, let $Neigh_d(u)$ is the set of all the $d-$ neighbours of node u. The maximum degree of a node in G is denoted as Δ and defined as: $\Delta = \max_{v \in V} |N(v)|$. It is to be noted that the nodes in this setting can be identified with $O(\log n)$ bits and this identifier is stored in a variable id.

In analyzing the performance of distributed algorithms, we generally use the definition of *rounds* [1]. In a single round, a node in a network can send/receive an arbitrary length message to/from one (or more) of its neighbours, and it can use the information received to perform any computational task.

3 BUILDING LOW STRETCH OVERLAY

In this section, we illustrate how to construct the low max stretch overlay G from the input graph $\hat{G}$. We propose to build the overlay $\hat{G}$ as illustrated in Algorithm 1.

We build the low max stretch overlay incrementally. The construction has been divided into three distinct phases. Firstly, we find a suitable center node that is ideally equidistant from all other nodes. Secondly, we construct a spanning tree that is rooted at the previous center node. This spanning tree of the underlay $\hat{G}$ is such that the distance between

Algorithm 1: BuildOverlay: Phases for constructing overlay G (executed in order)

(1) ***Locate Center***: Find a vertex u that is located "centrally" in the graph, from which all other nodes are *almost equidistant*. In other words, choose u such that $u = \text{argmax}_{u \in V}\{Closeness(u)\}$.
(2) ***Tree Build***: Construct a spanning tree T rooted at node u on the graph $\hat{G}$.
(3) ***Add Overlay Edges***: Reduce the max stretch of T from k to $\frac{2 \cdot k}{3}$ by adding a specific set X of $O(\frac{n}{k})$ edges from $\hat{G} \setminus T$. Call $G = T \cup X$ as the required low max stretch overlay of $\hat{G}$.

each pair of nodes is not very large. Such a spanning tree is already effective in reducing the max stretch criteria. In the last phase, we reduce the max stretch one step further by adding a set of edges to the tree. Albeit, the overlay does not retain the properties of a tree anymore due to these edges, but it effectively minimizes the max stretch value in G. We now take up each Phase individually and discuss in depth.

3.1 Locate Center - A heuristic to find the median center of $\hat{G}$

Algorithm 2: MedianCenter(u): Find an approximate median center of $\hat{G}$ at node u

Assumption Let τ be a spanning tree on $\hat{G}$ and R be the root of τ.
Execute BFS from u. Let X be the BFS tree.
for *each level dist in* X **do**
  $Neigh_dist(u) \leftarrow$ No. of nodes at level $dist$.
end
if u = *leaf node in* τ **then**
  Send token c to $Par(u)$ in τ.
end
if $u \neq$ *leaf node* $\wedge$ *token c received from all children of u in* τ **then**
  Send token c to $Par(u)$ in τ.
end
Wait for broadcast completion token from R in τ
Let $d \leftarrow 0$
while $d < |V|$ **do**
  Let $Count_d(u) \leftarrow |x : x \in Neigh_d(u)|$
  if $|Count_d(u)| \geq |V|/2$ **then**
    return u
  end
  $d \leftarrow d + 1$
end

Let us now look at a distributed algorithm to compute an approximate center of the graph that minimizes *Farness* and maximizes *Closeness* in $\hat{G}$. As stated earlier, we aim to find a vertex u such that $u = \text{argmax}_{u \in V}\{Closeness(u)\}$.We call such a node the *median center* [7]. We develop Algorithm 2 to find this median center that is approximately equidistant from all the remaining nodes in the network.

The distributed *MedianCenter* algorithm works as follows: first, construct a BFS tree from each node in $\hat{G}$. We populate a set $Neigh_{dist}$ for each node u that stores the number of neighbours of u at distances $dist = 1, 2, ..$ so on. In other words, $Neigh_{dist}(u)$ gives the count of the $dist-$ neighbours of u. After each node computes this value, we send a convergecast wave towards the root of the assumed spanning tree τ. The root R gets notified that all nodes in τ have computed the $Neigh_{dist}$ values, and it in turn send a broadcast wave indicating the completion of the pre-processing

phase. After receiving this completion token, each node u then finds the count of the number of neighbourhoods it falls under. If this count comes to be greater than or equal to half the number of nodes in $\hat{G}$, we declare u to be a median center. Note that this heuristic may return more than one median center in the graph. In such a case, we can choose the center node as the one with the highest *Count* value.

REFERENCES

[1] Alain Cournier, Stéphane Devismes, Franck Petit, and Vincent Villain. 2006. Snap-stabilizing depth-first search on arbitrary networks. *Comput. J.* 49, 3 (2006), 268–280.

[2] Narsingh Deo, Gurpur Prabhu, and Mukkai S Krishnamoorthy. 1982. Algorithms for generating fundamental cycles in a graph. *ACM Transactions on Mathematical Software (TOMS)* 8, 1 (1982), 26–42.

[3] Yuval Emek and David Peleg. 2009. Approximating minimum max-stretch spanning trees on unweighted graphs. *SIAM J. Comput.* 38, 5 (2009), 1761–1781.

[4] Roger C Entringer, Douglas E Jackson, and DA Snyder. 1976. Distance in graphs. *Czechoslovak Mathematical Journal* 26, 2 (1976), 283–296.

[5] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. 2015. Near-optimal distributed maximum flow. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. 81–90.

[6] Riko Jacob, Andrea Richa, Christian Scheideler, Stefan Schmid, and Hanjo Täubig. 2009. A distributed polylogarithmic time algorithm for self-stabilizing skip graphs. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*. 131–140.

[7] Edward Minieka. 1977. The centers and medians of a graph. *Operations Research* 25, 4 (1977), 641–650.

[8] Rizal Mohd Nor, Mikhail Nesterenko, and Christian Scheideler. 2013. Corona: A stabilizing deterministic message-passing skip list. *Theoretical Computer Science* 512 (2013), 119–129.

[9] Melih Onus, Andrea Richa, and Christian Scheideler. 2007. Linearization: Locally self-stabilizing sorting in graphs. In *2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 99–108.