Contents lists available at ScienceDirect
Discrete Applied Mathematics
journal homepage: www.elsevier.com/locate/dam

# Maintaining balanced trees for structured distributed streaming systems[☆]

F. Giroire[a,*], R. Modrzejewski[b], N. Nisse[c], S. Pérennes[a]

[a] Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
[b] Google, Dublin, Ireland
[c] Inria, France

A R T I C L E I N F O

*Article history:*
Received 11 May 2016
Received in revised form 30 June 2017
Accepted 3 July 2017
Available online xxxx

*Keywords:*
Distributed live streaming system
Graph algorithm
Balanced trees
Peer-to-peer

A B S T R A C T

In this paper, we propose and analyze a simple local algorithm to balance a tree. The motivation comes from live distributed streaming systems in which a source diffuses a content to peers via a tree, a node forwarding the data to its children. Such systems are subject to a high churn, peers frequently joining and leaving the system. It is thus crucial to be able to repair the diffusion tree to allow an efficient data distribution. In particular, due to bandwidth limitations, an efficient diffusion tree must ensure that node degrees are bounded. Moreover, to minimize the delay of the streaming, the depth of the diffusion tree must also be controlled. We propose here a simple distributed repair algorithm in which each node carries out local operations based on its degree and on the subtree sizes of its children. In a synchronous setting, we first prove that starting from any $n$-node tree our process converges to a balanced binary tree in $O(n^2)$ rounds. We then describe a more restrictive model, adding a small extra information to each node, under which we adapt our algorithm to converge in $\Theta(n \log n)$ rounds.

## 1. Introduction

Trees are inherent structures for data dissemination in general and particularly in peer-to-peer live streaming networks. As shown in [17], networks of this kind experience a high churn (rate of node joins and leaves). Leaves can be both graceful, where a node informs about imminent departure and network rearranges itself before it stops providing to the children, or abrupt (e.g. due to connection or hardware failure). In this case, the diffusion tree may be broken.

In this paper, we tackle this issue by designing an efficient maintenance scheme for trees. The problem setting is as follows. A single source provides live media to some nodes in the network. This source is the single reliable node of the network, all other peers may be subject to failure. Each node may relay the content to further nodes. Due to limited bandwidth, both source and any other node can provide media to a limited number $k \geq 2$ of nodes. The network is organized into a logical tree, rooted at the source of media. If node $x$ forwards the stream towards node $y$, then $x$ is the parent of $y$ in the logical tree. Note that the delay between broadcasting a piece of media by the source and receiving by a peer is given by its distance from the root in the logical tree. Hence, *our goal is to minimize the tree depth, while respecting degree constraints.*

In this work, we assume a *reconnection process*: when a node leaves, its children reattach to its parent. This can be done locally if each node stores the address of its grandparent in the tree. Note that this process is performed independently of the

☆ The research leading to these results has received funding from ANR program Investments for the Future under reference ANR-11-LABX-0031-01, Associated Team AlDyNet, project ECOS-Sud Chile and région PACA.

* Corresponding author.
*E-mail address:* frederic.giroire@cnrs.fr (F. Giroire).

http://dx.doi.org/10.1016/j.dam.2017.07.006
0166-218X/© 2017 Elsevier B.V. All rights reserved.

bandwidth constraint, hence after multiple failures, a node may become the parent of many nodes. This process can leave the tree in a state where either the bandwidth constraints are violated (the degree of a node is larger than $k$) or the tree depth is not optimal. Thus, we propose a distributed *balancing process*, where based on information about its degree and the subtree sizes of its children, a node may perform a local operation at each round. We show that this balancing process, starting from any tree, converges to a balanced tree and we provide analytic upper bounds of the convergence time. More precisely, our contributions are

- In Section 3, we provide a formal definition of the problem and *we propose a distributed algorithm for the balancing process*. The process works in a synchronous setting. At each round, all nodes are sequentially scheduled by an adversary and must execute the process.
- In Section 4, we show that the balancing process always succeeds in $O(n^2)$ rounds. Note that we prove a lower bound of $\Omega(n)$. Nevertheless, the proof of this result is non trivial, and, to the best of our knowledge, *this is the first theoretical analysis of the convergence time of a balancing process for live streaming systems.*
- Then, in Section 5, we study a restricted version of the algorithm in which a node performs an operation only when the subtrees of its children are balanced. In this case, we succeeded in obtaining a tight bound of $\Theta(n \log n)$ on the number of rounds for the worst tree.

The distributed algorithm is presented for any value of $k \geq 2$. However, the proofs of convergence are given for the case $k = 2$. We believe they can be extended to any value of $k$, but we leave this as future work, as the proofs for $k = 2$ are already long and tedious. A short version of this paper has previously been presented in the 20th Colloquium on Structural Information and Communication Complexity (SIROCCO) (Giroire et al., 2013 [14]).

## 2. Related work

**Live Streaming Systems.** Trees are inherent structures for data dissemination in general and particularly in peer-to-peer live streaming networks. Fundamentally, from the perspective of a peer, each atomic piece of content has to be received from some source and forwarded towards some receivers. Moreover, most of the actual streaming mechanisms ensure that a piece of information is not transmitted again to a peer that already possesses it. Therefore, this implies that dissemination of a single fragment defines a tree structure. Even in *unstructured* networks, whose main characteristic is lack of defined structure, many systems look into perpetuating such underlying trees, e.g. the second incarnation of Coolstreaming [17] or PRIME [19].

Unsurprisingly, early efforts into designing peer-to-peer video streaming concentrated on defining tree-based structures for data dissemination. These have been quickly deemed inadequate, due to fragility and unused bandwidth at the leaves of the tree. One possible fix to these weaknesses was introduced in SplitStream [7]. The proposed system maintains multiple concurrent trees to tolerate failures, and internal nodes in a tree are leaf nodes in all other trees to optimize bandwidth. The construction of intertwined trees can be simplified by a randomized process, as proposed in Chunkyspread [21], leading to a streaming algorithm performing better over a range of scenarios.

As found in [17], *node churn is the main difficulty for live streaming networks*, especially those trying to preserve structure. To overcome it, in [22], authors propose a stochastic optimization approach relying on constantly randomly creating and breaking relationships. To ensure network connectivity, nodes are said to keep open connections with hundreds of potential neighbors. This is usually not possible. Another approach, presented in [18], is churn-resiliency by maintaining redundancy within the network structure. Motivated by wireless sensors, authors of [20] face a similar problem of maintaining balanced trees, needed for connecting wireless sensors. However, their solution is periodical rebuilding the whole tree from scratch. *Our solution aims at minimizing the disturbance of nodes, whose ancestors were not affected by recent failures, as well as minimizing the redundancy in the network.*

Most of the analysis of these systems found in the literature focuses on the feasibility, construction time and properties of the established overlay network, see for example [7,21] and [8] for a theoretical analysis. But these works usually ignore the issue of tree maintenance. Generally, in these works, when some elements (nodes or links) of the networks fail, the nodes disconnected from the root execute the same procedure as for initial connection. To the best of our knowledge, there is *no theoretical analysis on the efficiency of tree maintenance in streaming systems*. The reliability usually is estimated by simulations or experiments as in [7].

**Self-stabilizing Algorithms.** The algorithm proposed in this paper aims at tolerating the high churn in a live distributed streaming system. In some extent, it is a fault-tolerant algorithm and it is natural to consider similar work in the context of self-stabilizing algorithms [11]. A self-stabilizing algorithm must ensure to reach a valid configuration (in our case, a $k$-balanced tree) starting from any initial configuration (where the tree may be arbitrary and the node's memory may be corrupted and unreliable).

The problem of spanning tree has been widely studied in this context since such a structure may be used for solving the leader election problem. Self-stabilizing algorithms for computing spanning trees have been proposed based on BFS tree [1,10], DFS tree [9], shortest path tree [2,6,16], spanning tree with minimum diameter [5], spanning tree with minimum maximum degree [4], etc. Some work has also been done in the context of Peer-to-Peer networks such as [15] that proposes a self-stabilizing algorithm for computing spanning tree in large scale systems where any pair of processes can communicate