Fast Distributed Computation in Dynamic Networks via Random Walks

Atish Das Sarma\textsuperscript{1}, Anisur Rahaman Molla\textsuperscript{2}, and Gopal Pandurangan\textsuperscript{3}

\textsuperscript{1} eBay Research Labs, eBay Inc., CA, USA
atish.dassarma@gmail.com
\textsuperscript{2} Division of Mathematical Sciences, Nanyang Technological University, Singapore 637371
anisurpm@gmail.com
\textsuperscript{3} Division of Mathematical Sciences, Nanyang Technological University, Singapore 637371 and Department of Computer Science, Brown University, Providence, RI 02912, USA
gopalpandurangan@gmail.com

Abstract. The paper investigates efficient distributed computation in dynamic networks in which the network topology changes (arbitrarily) from round to round. Random walks are a fundamental primitive in a wide variety of network applications; the local and lightweight nature of random walks is especially useful for providing uniform and efficient solutions to distributed control of dynamic networks. Given their applicability in dynamic networks, we focus on developing fast distributed algorithms for performing random walks in such networks.

Our first contribution is a rigorous framework for design and analysis of distributed random walk algorithms in dynamic networks. We then develop a fast distributed random walk based algorithm that runs in $\tilde{O}(\sqrt{\tau \Phi})$ rounds\textsuperscript{1} (with high probability), where $\tau$ is the dynamic mixing time and $\Phi$ is the dynamic diameter of the network respectively, and returns a sample close to a suitably defined stationary distribution of the dynamic network.

Our next contribution is a fast distributed algorithm for the fundamental problem of information dissemination (also called as gossip) in a dynamic network. In gossip, or more generally, $k$-gossip, there are $k$ pieces of information (or tokens) that are initially present in some nodes and the problem is to disseminate the $k$ tokens to all nodes. We present a random-walk based algorithm that runs in $\tilde{O}(\min\{n^{1/3}k^{2/3}(\tau \Phi)^{1/3}, nk\})$ rounds (with high probability). To the best of our knowledge, this is the first $o(nk)$-time fully-distributed token forwarding algorithm that improves over the previous-best $O(nk)$ round distributed algorithm [Kuhn et al., STOC 2010], although in an oblivious adversary model.

Keywords: Dynamic Network, Distributed Algorithm, Random walks, Random sampling, Information Dissemination, Gossip.

\textsuperscript{*} Supported in part by the following research grants: Nanyang Technological University grant M58110000, Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2010-T2-2-082, and a grant from the US-Israel Binational Science Foundation.

\textsuperscript{1} $\tilde{O}$ hides polylog $n$ factors where $n$ is the number of nodes in the network.
1 Introduction

Random walks play a central role in computer science spanning a wide range of areas in both theory and practice. Random walks are used as an integral subroutine in a wide variety of network applications ranging from token management and load balancing to search, routing, information propagation and gathering, network topology construction and building random spanning trees (e.g., see [11] and the references therein). They are particularly useful in providing uniform and efficient solutions to distributed control of dynamic networks [6,23]. Random walks are local and lightweight and require little index or state maintenance which make them especially attractive to self-organizing dynamic networks such as peer-to-peer, overlay, and ad hoc wireless networks. In fact, in highly dynamic networks, where the topology can change arbitrarily from round to round (as assumed in this paper), extensive distributed algorithmic techniques that have been developed for the last few decades for static networks (see e.g., [21,16,22]) are not readily applicable. On the other hand, we would like distributed algorithms to work correctly and terminate even in networks that keep changing continuously over time (not assuming any eventual stabilization). Random walks being so simple and very local (each subsequent step in the walk depends only on the neighbors of the current node and does not depend on the topological changes taking place elsewhere in the network) can serve as a powerful tool to design distributed algorithms for such highly dynamic networks. However, it remains a challenge to show that one can indeed use random walks to solve non-trivial distributed computation problems efficiently in such networks, with provable guarantees. Our paper is a step in this direction.

A key purpose of random walks in many of the network applications is to perform node sampling. While the sampling requirements in different applications vary, whenever a true sample is required from a random walk of certain steps, typically all applications perform the walk naively — by simply passing a token from one node to its neighbor: thus to perform a random walk of length \( \ell \) takes time linear in \( \ell \). In prior work [11,12], the problem of performing random walks in time that is significantly faster, i.e., sublinear in \( \ell \), was studied. In [12], a fast distributed random walk algorithm was presented that ran in time sublinear in \( \ell \), i.e., in \( \tilde{O}(\sqrt{D}) \) rounds (where \( D \) is the network diameter). This algorithm used only small sized messages (i.e., it assumed the standard CONGEST model of distributed computing [21]). However, a main drawback of this result is that it applied only to static networks. A major problem left open in [12] is whether a similar approach can be used to speed up random walks in dynamic networks.

The goals of this paper are two fold: (1) giving fast distributed algorithms for performing random walk sampling efficiently in dynamic networks, and (2) applying random walks as a key subroutine to solve non-trivial distributed computation problems in dynamic networks. Towards the first goal, we first present a rigorous framework for studying random walks in a dynamic network (cf. Section 2). (This is necessary, since it is not immediately obvious what the output of random walk sampling in a changing network means.) The main purpose of our random walk algorithm is to output a random sample close to the “stationary