Theoretical Computer Science 581 (2015) 45–66

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Distributed computation in dynamic networks via random walks ☆

Atish Das Sarma [a], Anisur Rahaman Molla [b,*], Gopal Pandurangan [c,1]

[a] eBay Research Labs, eBay Inc., CA, USA
[b] Division of Mathematical Sciences, Nanyang Technological University, Singapore 637371, Singapore
[c] Department of Computer Science, University of Houston, Houston, TX, 77204 USA

A R T I C L E   I N F O

Article history:
Received 28 August 2013
Received in revised form 26 February 2015
Accepted 27 February 2015
Available online 6 March 2015
Communicated by D. Peleg

Keywords:
Dynamic network
Distributed algorithm
Random walks
Random sampling
Information dissemination
Gossip

A B S T R A C T

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 the 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[2] (with high probability), where τ is the *dynamic mixing time* and Φ 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 $\tilde{O}(\min\{n^{1/3}k^{2/3}(\tau\Phi)^{1/3}, k\Phi\})$ rounds (with high probability). This is the first $o(k\Phi)$-time fully-distributed *token forwarding* algorithm (on certain graph families) that improves over the previous-best $O(k\Phi)$ round distributed algorithm (Kuhn et al. [24]), although in an oblivious adversary model.

The random walk framework developed in this paper has also subsequently proved useful in developing efficient storage and search algorithms as well as developing fast byzantine agreement algorithms in dynamic networks.

☆ A preliminary version of the paper appeared in the proceedings of 26th International Symposium on Distributed Computing (DISC), volume 7611 of LNCS, Springer, pp. 136–150, 2012 [16].

* Corresponding author.
E-mail addresses: atish.dassarma@gmail.com (A. Das Sarma), anisurpm@gmail.com (A.R. Molla), gopalpandurangan@gmail.com (G. Pandurangan).

1 Supported in part by the following research grants: Nanyang Technological University grant M58110000, Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 2 grant MOE2010-T2-2-082, Singapore MOE AcRF Tier 1 grant MOE2012-T1-001-094, and grant 2008348 from the US–Israel Binational Science Foundation (BSF). Research done while at Division of Mathematical Sciences, Nanyang Technological University, Singapore 637371.

2 $\tilde{O}$ hides polylog n factors where n is the number of nodes in the network.

http://dx.doi.org/10.1016/j.tcs.2015.02.044
0304-3975/

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 [18] and the references therein). They are particularly useful in providing uniform and efficient solutions to distributed control of dynamic networks [11,36]. Random walks are local and lightweight and require little index or state maintenance which makes 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., [27,34,35]) 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 is challenging 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 ℓ takes time linear in ℓ. In prior work [18], the problem of performing random walks in time that is significantly faster, i.e., sublinear in ℓ, was studied. In [18], a fast distributed random walk algorithm was presented that ran in time sublinear in ℓ, i.e., in $\tilde{O}(\sqrt{\ell D})$ 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 [34]). However, a main drawback of this result is that it applied only to *static* networks. A major problem left open in [18] is whether a similar approach can be used to speed up random walks in dynamic networks.

The goals of this paper are twofold: (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 distribution" (defined formally only in the case of *regular dynamic network* in Section 2) of the underlying dynamic network. Our random walk algorithms work under an oblivious adversary that fully controls the dynamic network topology, but does not know the random choices made by the algorithms (cf. Section 3 for precise problem statements and results). We present a fast distributed random walk algorithm that runs in $\tilde{O}(\sqrt{\tau\Phi})$ with high probability (w.h.p.),[3] where τ is (an upper bound on) the dynamic mixing time and Φ is the dynamic diameter of the network respectively (cf. Section 6). Our algorithm uses small-sized messages only and returns a node sample that is "close" to the stationary distribution of the dynamic network (assuming the stationary distribution remains fixed even as the network changes). (The precise definitions of these terms are deferred to Section 2.) We further extend our algorithm to efficiently perform and return κ independent random walk samples in $\tilde{O}(\min\{\sqrt{\kappa\tau\Phi}, \kappa+\tau\})$ rounds (cf. Section 7). This is directly useful in the application considered in this paper.

Towards the second goal, we present a key application of our fast random walk sampling algorithm (cf. Section 8). We present 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. In an n-node network, solving n-gossip allows nodes to distributively compute any computable function of their initial inputs using messages of size $O(\log n + d)$, where d is the size of the input to the single node [24]. We present a random-walk based algorithm that runs in $\tilde{O}(\min\{n^{1/3}k^{2/3}(\tau\Phi)^{1/3}, k\Phi\})$ rounds with high probability. On certain graph families, this gives the first $o(k\Phi)$-time fully-distributed *token forwarding* algorithm that improves over the previous-best $O(k\Phi)$ round distributed algorithm [24],[4] albeit under an oblivious adversarial model. A lower bound of $\Omega(nk/\log n)$ under the adaptive adversarial model of [24], was recently shown in [21]; hence one cannot do substantially better than the $O(nk)$ algorithm in general under an adaptive adversary.

The random walk framework developed in this paper has also subsequently proved useful in design and analysis of efficient distributed algorithms in dynamic networks for other problems as well (cf. Section 4.1): (1) developing efficient storage and search algorithms in dynamic networks [2] and (2) developing fast byzantine agreement algorithms in dynamic networks [3].

[3] With high probability means with probability at least $1-1/n^{\Omega(1)}$, where n is the number of nodes in the network.

[4] This bound in Kuhn et al. [24] paper is given in terms of $O(kn)$, since the dynamic diameter Φ can be at most n (cf. Section 2.2).