Systems & Control Letters 92 (2016) 20–27

Contents lists available at ScienceDirect

Systems & Control Letters

journal homepage: www.elsevier.com/locate/sysconle

Distributed finite-time calculation of node eccentricities, graph radius and graph diameter

Gabriele Oliva [a,*], Roberto Setola [a], Christoforos N. Hadjicostis [b]

[a] Unit of Automatic, Department of Engineering, Università Campus Bio-Medico di Roma, via Álvaro del Portillo 21, 00128, Rome, Italy

[b] Department of Electrical and Computer Engineering, University of Cyprus, 75 Kallipoleos Avenue, P.O. Box 20537, 1678 Nicosia, Cyprus

ARTICLE INFO

Article history:
Received 9 July 2015
Received in revised form
25 November 2015
Accepted 18 February 2016
Available online 24 March 2016

Keywords:
Distributed algorithms
Diameter
Eccentricity
Radius
Max-consensus

ABSTRACT

The distributed calculation of node eccentricities, graph radius and graph diameter are fundamental steps to tune network protocols (e.g., setting an adequate time-to-live of packets), to select cluster heads, or to execute distributed algorithms, which typically depend on these parameters. Most existing methods deal with undirected topologies and have high memory and/or bandwidth requirements (or simply provide a bound on the diameter to reduce such costs). Other approaches, instead, require nodes able to communicate with their neighbors on a point-to-point basis, thus requiring each node to be aware of its neighbors. In this paper, we consider strongly connected directed graphs or connected undirected graphs, and we develop an algorithm that takes advantage of the robustness and versatility of the max-consensus algorithm, and has low computational, memory and bandwidth requirements. Moreover, the agents communicate by broadcasting messages to their (out-) neighbors without requiring any knowledge on them or needing point-to-point communication capability. Specifically, each node has memory occupation proportional to the number of its neighbors, while the bandwidth for each link at each time instant is $O(\log n)$ bits, where n is the number of nodes. The completion time of the proposed algorithm is $O(\delta n)$ for undirected graphs and $O(n^2)$ for directed graphs, where δ is the network diameter.

1. Introduction

In the literature much effort has been spent in the parallel (see [1] for a recent survey) and distributed calculation of the network diameter (i.e., the length of the maximum shortest path among any two nodes in the network), of the eccentricity of a given node (i.e., the maximum shortest path from a particular node to any other node) and the network radius (i.e., the minimum among the eccentricities of the nodes). Having such insights can help reduce computational effort in consensus algorithms [2–4], or can be used to set time-to-live parameters in routing protocols [5]. Moreover, the eccentricities of the nodes can be used to select cluster heads or local coordinators [6]. In the following, we denote the total number of nodes/agents in the network by n, and the network diameter by δ; we also denote by $|\mathcal{N}_i^{in}|$ the size of the in-neighborhood $\mathcal{N}_i^{in}$ of an agent i, i.e., the number of agents that may send information to the ith agent.

The approaches available to date apply to the connected undirected graph case and, although some methods like the one in [7] have short completion time, they may have high memory and/or bandwidth requirements, which become overwhelming, especially for big networks. A typical approach to calculate eccentricities [6,8] is to construct the minimum spanning tree [9] rooted at each node, or to calculate the shortest paths in the graph [10,11]. For undirected graphs, a distributed way to calculate the shortest paths is given in [12] and it requires $O(n^2)$ steps to complete.

The approach for undirected graphs presented in [13], is able to calculate the diameter in $O(n)$ steps, $O(\log n)$ bandwidth per link per step and $O(|\mathcal{N}_i^{in}| \log n)$ bits of memory at each node (if a time-to-live is attached to the messages, otherwise $O(n \log n)$ bits are required to check for retransmissions). The algorithm in [13] requires nodes outfitted with nontrivial communication and computational capabilities, as it combines several communication approaches, ranging from flooding and breadth-first visit (broadcast) to depth-first visit and ConvergeCast (point-to-point). Specifically, the algorithm calculates the maximum among the eccentricities of the nodes by constructing n trees via a breadth-first visit; the visits are suitably staggered to avoid collisions and terminate in $O(n)$ steps. This feature, however, relies on a precise scheduling of the

* Corresponding author.
E-mail addresses: g.oliva@unicampus.it (G. Oliva), r.setola@unicampus.it (R. Setola), chadjic@ucy.ac.cy (C.N. Hadjicostis).

http://dx.doi.org/10.1016/j.sysconle.2016.02.015
0167-6911/© 2016 Elsevier B.V. All rights reserved.

Table 1
Comparison of the main features of the proposed algorithm against the state of the art.

Feature	Ref. [13]	Ref. [14]	Ref. [7]	Ref. [15]	Ref. [4]	Proposed approach
Approximation	Exact	$\delta \le \overline{\delta} \le 2\delta$	Exact	$\delta \le \overline{\delta} \le (\frac{3}{2}+\eta)\delta$	$\delta \le \overline{\delta} \le 2\delta$	Exact
Undirected	Yes	Yes	Yes	Yes	Yes	Yes
Directed	No	No	No	No	No	Yes
Steps	$O(n)$	$O(n)$	$O(\delta)$	$O(\sqrt{\frac{n\log n}{\delta\eta}}+\delta)$	$O(\delta n)$	$O(n^2)$ for directed graphs, $O(\delta n)$ for undirected graphs
Memory	$O(\lvert\mathcal{N}_i^{in}\rvert \log n)$ if using a time-to-live approach for Flooding, $O(n\log n)$ if checking for re-transmission	$O(M\lvert\mathcal{N}_i^{in}\rvert \log n)$	$O(n\log n)$	$O(\lvert\mathcal{N}_i^{in}\rvert \log n)$	$O(\lvert\mathcal{N}_i^{in}\rvert \log n)$	$O(\lvert\mathcal{N}_i^{in}\rvert \log n)$
Bandwidth	$O(\log n)$	$O(M\log n)$	$O(n\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
Remarks	The nodes must be able to switch between broadcasting and point-to-point communication. The algorithm requires a precise scheduling of the messages exchanged.	$M \ge 1$ is used to trade off memory/bandwidth for accuracy		$\eta \in (0, \frac{1}{3}]$ is used to trade off completion time for accuracy		

visits, and a failure (e.g., packet loss) may cause errors or inconsistencies.

In [14] a procedure based on max-consensus [2] is used in order to obtain an approximation $\overline{\delta}$ of the diameter δ in $O(n)$ steps; the accuracy, however, depends on a factor M which also influences memory requirements ($O(M|\mathcal{N}_i^{in}|\log n)$ bits for each agent) and bandwidth ($O(M\log n)$ bits per link per time step). In [7] an algorithm for the exact calculation of the eccentricity of each node and of the network diameter is provided. Specifically, the nodes exchange messages containing information on an estimate of the diameter, on the hop counts and on the identifier of the sender, following a flooding approach. The algorithm requires a bandwidth of $O(n\log n)$ bits per link per step and terminates in $O(\delta)$ steps. As for the memory resources, at each step, the agents need to store the identifiers associated to the messages received in the previous 2 steps, hence it requires storage of $O(n)$ identifiers (i.e., $O(n\log n)$ bits of memory occupancy). In [15] the authors develop an algorithm that provides an approximation $\overline{\delta}$ of the diameter δ such that $\overline{\delta} \le (\frac{3}{2}+\eta)\delta$, where $\eta \in (0, \frac{1}{3}]$ is a parameter that constitutes a trade off between accuracy and completion time. Each node can transmit a different message of $O(\log n)$ bits to each of its neighbors in a synchronous way, and the graph is assumed to be undirected. The number of rounds for the algorithm to terminate is $O(\sqrt{\frac{n\log n}{\delta\eta}}+\delta)$.

In [4] we develop an algorithm that provides an upper bound of the diameter in $O(\delta n)$ steps; this upper bound is guaranteed to be at most twice the actual diameter. The memory requirement for this algorithm is $O(|\mathcal{N}_i^{in}|\log n)$ bits per node and the bandwidth is $O(\log n)$ bits per link per step.

In this paper, we provide a distributed algorithm which provides the exact value of the eccentricities, graph radius and graph diameter, both in the undirected and directed graph cases. The proposed algorithm maintains low bandwidth (i.e., $O(\log n)$ bits per link per step) and memory (i.e., $O(|\mathcal{N}_i^{in}|\log n)$ bits per node). Moreover, in the proposed approach each node only has to broadcast its messages, without any knowledge of its neighbors and without requiring point-to-point communication capability. In a nutshell, the algorithm calculates the eccentricities of the nodes in a sequential way, resorting to max-consensus algorithms [2]. Notice that the usage of max-consensus provides increased robustness to transmission failures such as packet loss (although mainly related to the average consensus case, some hints on this issue can be found in [16–18]), compared to more fragile approaches like the one in [13].

As for the completion time, the proposed algorithm terminates in $O(\delta n)$ steps in the connected and undirected graph case and $O(n^2)$ steps in the strongly connected and directed graph case. Table 1 summarizes the comparison between the proposed approach and the state of the art.

The proposed algorithm relies on successive runs of the max-consensus algorithm, and on a novel algorithm to calculate the depth of each node over the minimum spanning tree rooted at a given node, which has a structure similar to max-consensus. This approach has several advantages in terms of memory, bandwidth and robustness when compared against previous approaches. The advantages are explained in detail later on, once we have the chance to more precisely describe the characteristics of the algorithm.

The outline of the paper is as follows: Section 2 collects some background material; Sections 3 and 4 develop our algorithms to calculate the eccentricity and the diameter (and radius), respectively; Section 5 contains simulations that illustrate the potential of the proposed approach. Finally, some conclusive remarks and future work directions are collected in Section 6.

2. Preliminaries

Let $G = \{V, E\}$ be a *graph* with n nodes $V = \{v_1, v_2, \dots, v_n\}$ and e edges $E \subseteq V \times V$, where $(v_i, v_j) \in E$ captures the existence of a link from node v_i to node v_j. A graph is said to be *undirected* if $(v_i, v_j) \in E$ whenever $(v_j, v_i) \in E$, and is said to be *directed* otherwise.

A *path* over a graph $G = \{V, E\}$, starting from a node $v_i \in V$ and ending in a node $v_j \in V$, is a subset of links in E that connect v_i and v_j without creating loops. The *length* of the path is the cardinality of such set. A graph is *connected* if for each pair of nodes v_i, v_j there is a path over G that connects them without necessarily respecting the edge orientation, while it is *strongly connected* if the path respects the orientation of the edges. It follows that every undirected connected graph is also strongly connected.

A *minimum path* that connects v_i and v_j is the path from v_i to v_j of minimum length. A *minimum spanning tree* rooted at a node $v_i \in V$ is a *tree* (i.e., an acyclic connected subgraph of G with $n-1$ links, where n is the number of nodes) that connects v_i to each other node via edges belonging to the minimum paths in G