Fast Distributed Computation of Distances in Networks

Paulo Sérgio Almeida, Carlos Baquero, Alcino Cunha

psa@di.uminho.pt, cbm@di.uminho.pt, alcino@di.uminho.pt

Techn. Report DI-CCTC-10-11

2010, September

Computer Science and Technology Center
Departamento de Informática da Universidade do Minho
Campus de Gualtar – Braga – Portugal
http://cctc.di.uminho.pt/

DI-CCTC-10-11
Fast Distributed Computation of Distances in Networks
by Paulo Sérgio Almeida, Carlos Baquero, Alcino Cunha

Abstract

This paper presents a distributed algorithm to simultaneously compute the diameter, radius and node eccentricity in all nodes of a synchronous network. Such topological information may be useful as input to configure other algorithms. Previous approaches have been modular, progressing in sequential phases using building blocks such as BFS tree construction, thus incurring in longer executions than strictly required. We present an algorithm that, by timely propagation of available estimations, achieves a faster convergence to the correct values. We show local criteria for detecting convergence in each node. The algorithm avoids the creation of BFS trees and simply manipulates sets of node ids and hop counts. For the worst scenario of variable start times, each node i with eccentricity $\mathrm{ecc}(i)$ can compute: the node eccentricity in $D + \mathrm{ecc}(i) + 2$ rounds; the diameter in $2D + \mathrm{ecc}(i) + 2$ rounds; and the radius in $D + \mathrm{ecc}(i) + 2R$ rounds.

1 Introduction

This paper presents a distributed algorithm to simultaneously compute the diameter, radius and node eccentricity in all nodes of a network. An early knowledge of this topological information is useful since it is often used as input to other algorithms. For instance, the diameter or eccentricity can be used to simplify termination in leader election algorithms [8] and calibrate *time-to-live* parameters [7]; the radius and eccentricity allow determining center nodes [6], which are nice candidates to serve as coordinators in other distributed algorithms.

We assume a synchronous network model, while allowing variable start times, in which one or more nodes can start the algorithm with no prior coordination. The algorithm is designed to be fast in a precise sense; we are concerned with, not just asymptotic complexity, but exact bounds in the number of rounds.

The classic approach to this problem [8] is to compute the eccentricities by parallel construction of *breadth first search* (BFS) trees rooted at each node. Once eccentricities are known, each BFS tree can be reused to do a global computation, starting from the leafs and converging to each root node, allowing each to compute the maximum and minimum eccentricity (the network diameter D and radius R). Considering a graph $G = (V, E)$, this classic approach has message complexity of $\Theta(|V|\,|E| \log |V|)$ bits. The diameter and radius are known at all nodes in at most $4D + 2$ rounds.

These time bounds can be improved if one departs from this modular multi-phase approach, where BFS trees are first constructed to compute eccentricities. This paper introduces an algorithm that propagates candidate values in a timely and continuous fashion, resulting in a faster convergence to the correct values. The challenge in this strategy is that a suitable termination method must be devised to detect, in each node, when the candidate values have converged. Under the same message complexity of the classic approach, the proposed algorithm reduces the number of rounds to compute the diameter to at most $3D + 1$ rounds and the radius to at most $2D + 2R$ rounds. To be more precise, with this algorithm each node i with eccentricity $\mathrm{ecc}(i)$ computes:

- the node eccentricity at most by round $D + \mathrm{ecc}(i) + 2$;
- the diameter at most by round $2D + \mathrm{ecc}(i) + 2$; and
- the radius at most by round $D + \mathrm{ecc}(i) + 2R$.

The paper is organized as follows. Section 2 presents the computing model and introduces notation. The algorithm is presented in Section 3. Example runs of the algorithm, proofs of local convergence criteria and global convergence bounds are also included in this section. The related work is discussed in Section 4, and the conclusions are presented in Section 5.

2 Network Model and Notation

We assume a synchronous network model, similar to the one described in [8]. The network is composed by a set of nodes connected by links, which we assume to be bidirectional; i.e., we have a simple, connected, unweighted, undirected graph $G = (V, E)$ with $|V| \geq 2$ nodes and $|E| \geq 1$ links. We assume globally unique identifiers for nodes, but no knowledge of the network topology or the number of nodes.

Computation proceeds in synchronous rounds. At each round, nodes first look at their state and compute what messages are sent, through a *message-generation function*; then nodes look at their state and messages received and compute the new state after the round, through a *state-transition function*. We assume no link or process failures. In order to obtain an asynchronous version of the algorithm a synchronizer α [1] can be used. We operate under a maximum bandwidth of $O(|V| \log |V|)$ bits, per link per round.

We assume the general case with variable start times. Nodes start as *quiescent*, a state in which they do not send messages nor transition to different states. Nodes wakeup when they receive a message from a special *environment node* (not part of G, connected to every node), or from an already *active* node.

state variables:

e_i, node eccentricity, initially $e_i = 0$
d_i, network diameter, initially $d_i = 0$
r_i, network radius, initially $r_i = \infty$
s_i, status, initially $s_i = \text{QUIESCENT}$
I_i, set of node ids, initially $I_i = \{\}$
c_i, consecutive rounds with no new BFS, initially $c_i = 0$
O_i, message to be sent, initially $O_i = \{\}$

message-generation function:

$\text{msg}_i(\langle e_i, d_i, r_i, s_i, I_i, c_i, O_i\rangle, j) = O_i \qquad j \in \text{nbrs}(i)$

state-transition function:

$\text{trans}_i(\langle e_i, d_i, r_i, s_i, I_i, c_i, O_i\rangle, M_i) = \langle e'_i, d'_i, r'_i, s'_i, I'_i, c'_i, O'_i\rangle$

where

$M = \bigcup\{m \mid m \in M_i\}$
if $s_i = \text{QUIESCENT} \wedge M = \{\}$ **then**
  $\langle e'_i, d'_i, r'_i, s'_i, I'_i, c'_i, O'_i\rangle = \langle e_i, d_i, r_i, s_i, I_i, c_i, O_i\rangle$
else
  $M' = \{\langle \text{BFS}, j, h+1\rangle \mid \langle \text{BFS}, j, h\rangle \in M, j \notin I_i\}$
  $M'' = M' \cup \{\langle \text{BFS}, i, 0\rangle \mid s_i = \text{QUIESCENT}\}$
  if $M'' = \{\}$ **then**
    $c'_i = c_i + 1$
  else
    $c'_i = 0$
  $e'_i = \max(\{e_i\} \cup \{h \mid \langle \text{BFS}, \_, h\rangle \in M'\})$
  $d'_i = \max(\{d_i\} \cup \{d \mid \langle \text{DIAM}, d\rangle \in M\} \cup \{e'_i\})$
  $r'_i = \min(\{r_i\} \cup \{r \mid \langle \text{RAD}, r\rangle \in M\} \cup \{e'_i \mid c'_i = 2\})$
  $s'_i = \text{ACTIVE}$
  $I'_i = I_i \cup \{j \mid \langle \text{BFS}, j, \_\rangle \in M''\}$
  $M^d = \{\langle \text{DIAM}, d'_i\rangle \mid d'_i > d_i\}$
  $M^r = \{\langle \text{RAD}, r'_i\rangle \mid r'_i < r_i\}$
  $O'_i = M'' \cup M^d \cup M^r$

Figure 1: Algorithm.

We use $\text{d}(i, j)$ to denote the distance between nodes i and j (the length of the shortest path between nodes i and j); $\text{ecc}(i)$ for the eccentricity of node i (the maximum $\text{d}(i, j)$ between node i and any other node j); D for the network diameter (the maximum eccentricity over all nodes); R for the network radius (the minimum eccentricity over all nodes); and $\text{nbrs}(i)$ for the set of neighbors of node i (nodes connected to node i by a link).

3 Algorithm

The algorithm is presented in Figure 1. At each round, a node i sends the same message to all its neighbors (state variable O_i). A message is a non-empty set of tuples; the empty set represents absence of a message. The tuples in a message can be $\langle \text{BFS}, \_, \_\rangle$, $\langle \text{DIAM}, \_\rangle$ or $\langle \text{RAD}, \_\rangle$, where BFS, DIAM and RAD are constants. Nodes do not need to distinguish between messages that arrive from different neighbors; the second parameter of the state-transition function (parameter M_i) is the set of messages received by node i from all neighbors.

Figure 2: Path graph.

Each node when awoken broadcasts a BFS message with its id and a hop counter, which starts at 0. Nodes keep the set of ids of all received BFS messages. When a node receives a BFS message from a node not yet known, it increments the hop counter and rebroadcasts it.

Nodes know their eccentricity is at least the largest hop count received in a BFS message, which they keep in a variable (e_i). Nodes also keep in two other variables (d_i, r_i) a lower bound estimate for the diameter and an upper bound estimate for the radius. When a node increases the diameter estimate; it broadcasts a DIAM message with the new value. A node increases its diameter value when (1) its eccentricity surpasses its diameter estimate or (2) it receives a DIAM message whose value is higher that its diameter value. Estimation of the radius is also driven by eccentricity, but must be deferred until each node detects its correct eccentricity. When that happens, and also when a lower estimate for the radius is received, a RAD message is broadcast.

It is easy to see that at some point every node will have awakened; later everyone will have received a BFS from everyone else and will have their eccentricity stored in the respective variable; and later still nodes will receive some DIAM message with the network diameter, originating from some maximum eccentricity node (a periphery node). Similarly, a RAD message originating from a minimum eccentricity node (a center node) will arrive eventually at all nodes.

The relevant question is convergence detection, i.e., when will nodes know that their eccentricity, diameter and radius variables have converged to the correct values. For this purpose, and inspired by the approach in [9], nodes have a variable which stores the number of consecutive rounds for which no new BFS messages arrived. Later, we will show how this variable can be used for convergence detection.

In order to analyze the communication complexity of the whole execution of the algorithm, we can first observe that each BFS message can be encoded in $\Theta(\log |V|)$ bits, since its dominated by the size of the ids. Each node retransmits exactly one BFS message for every other node, totaling $\Theta(|V|\,|E| \log |V|)$ bits. Since the diameter can only increase at most D times, each node broadcasts at most D DIAM messages, totaling $O(D\,|E| \log D)$ bits. Similarly for RAD messages. Thus, total message complexity is $\Theta(|V|\,|E| \log |V|)$ bits.

3.1 Example Runs

Prior to the formal proofs of the algorithm properties, we now convey some intuition by illustrating its execution in two different graphs. The first graph is a path with eleven nodes, depicted in Figure 2. Nodes 0 and 10 and are the only two nodes in the periphery, thus defining the diameter; node 5 is the single node in the graph center. We consider a run where node 0 is activated and we will observe how the local variables evolve in nodes 0, 5 and 10 (Figures 3, 4 and 5, respectively).

When probing the variables at node 0 we can observe that each two rounds a new BFS is received until BFS messages from all nodes have arrived. Each time a new BFS is received the c_i counter is reset. Thus, when c_i reaches two the node knows that its eccentricity variable e_i has reached the correct final value. This happens in nodes 0, 5 and 10 at rounds 22, 17 and 22, respectively.

Once eccentricity is stabilized nodes start disseminating RAD messages in order to detect the minimum eccentricity. Since local radius estimates can only decrease, one must establish a termination condition so that each node knows that it reached the correct radius. We will show that this is safely achieved when $c_i \geq 2r_i$. The run in Figure 5 shows that no sooner than this has the radius reached its final value of 5 in node 10.

Diameter dissemination, via DIAM messages, starts even if nodes are still updating their (monotonically increasing) estimates for eccentricity. For instance, in Figure 4 we see that node 5 is activated at round 5 (receiving BFS messages from nodes 0 to 4) and sets both e_i and d_i to 5 (its

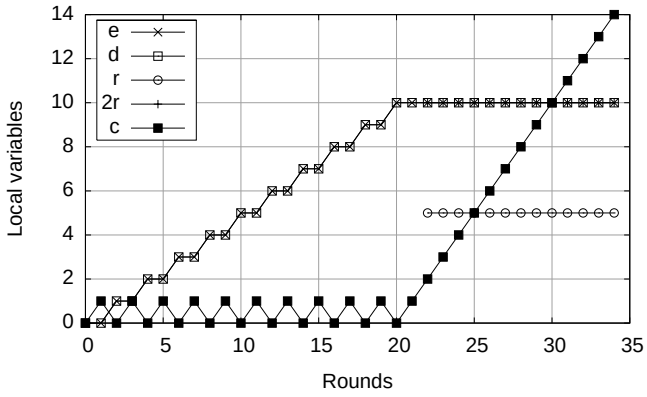

Figure 3: Path graph, start at 0, probing at node 0.

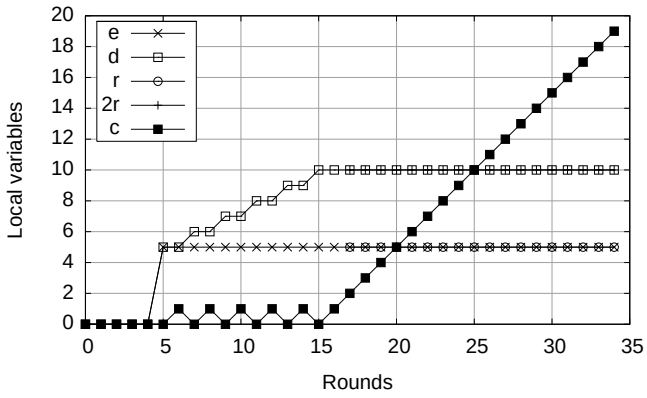

Figure 4: Path graph, start at 0, probing at node 5.

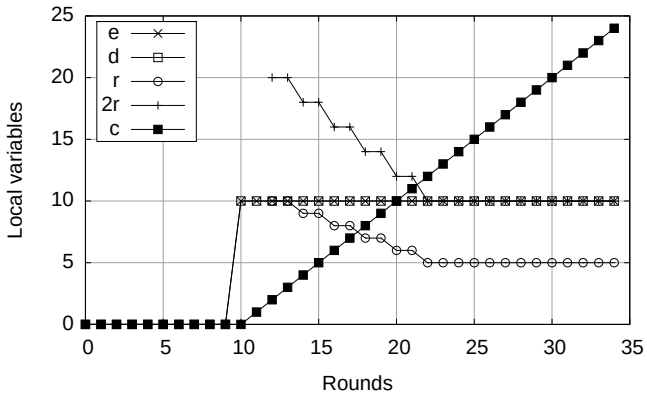

Figure 5: Path graph, start at 0, probing at node 10.

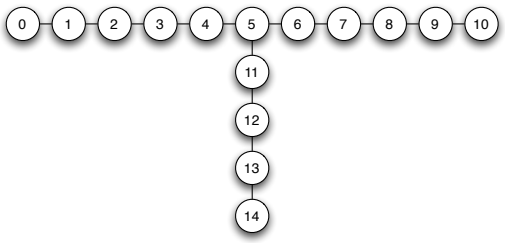

Figure 6: T shaped graph.

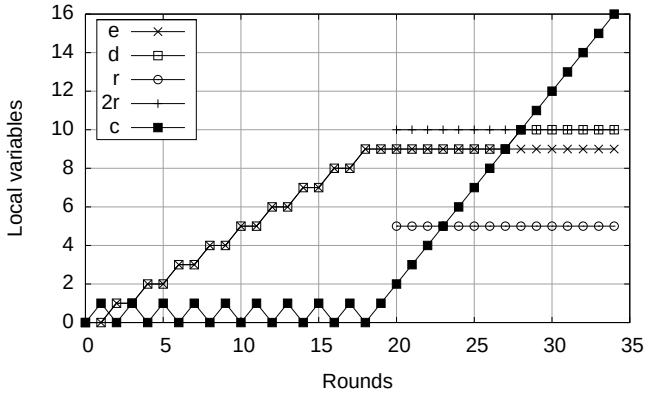

Figure 7: T shaped graph, start at 14, probing at node 14.

received hop distance from node 0, the furthest away). As nodes 6 to 10 are activated, in the next rounds, the diameter estimate d_i at node 5 gets updated each two rounds. We will show that in order for a node to know that the diameter has reached its final value it needs to locally observe that $c_i \geq 2$ and $c_i > d_i$. This happens in nodes 0, 5 and 10 at rounds 31, 26 and 21, respectively. One can notice that in this path graph the diameter is stable even before those rounds.

However, it is easy to construct a T shaped graph where detection cannot occur before the mentioned round. In Figure 6 we show such a graph. Here, a critical case happens when a node not in a diameter defining path (for example, node 14) is the first to be activated. As seen in Figure 7 this leads to a run where the diameter estimation seems to be stable for a large number of rounds (between round 18 and 27), finally increases at round 28, with convergence being detected only when $c_i > d_i$ at round 29. In the next section we prove that these convergence criteria are correct for general graphs.

3.2 Local Convergence Criteria

We now establish results that allow a node to know, using local information, that the variables estimating node eccentricity, network diameter and network radius have converged to the correct values. In the following we will use i, j, u, and v to range over node ids and r, n and k to range over non-negative integers.

Definition 3.1. $A_i(r) \doteq \{j \mid \mathrm{d}(i,j) \leq r \wedge s_j(r - \mathrm{d}(i,j)) = \textsc{active}\}$.

$A_i(r)$ denotes the "area of visibility" of node i after[1] round r: the set of nodes whose BFS messages arrive at i no later than round r. It is easy to see that A_i is monotonic: $A_i(r) \subseteq A_i(r+1)$.

[1]When referring to state variables, we do not use the expression "at round r" to avoid ambiguity between beginning or end of round. Throughout the paper we use "after round r" as a shorthand for "when round r has finished", i.e., "at the end of round r".

Lemma 3.1. *After any round r, $I_i(r) = A_i(r)$.*

Proof. If $j \in I_i(r)$ then i must have received not later than round r a BFS message starting at j; it traveled $\mathrm{d}(i,j)$ hops, which means that $\mathrm{d}(i,j) \leq r$ and j was active after round $r - \mathrm{d}(i,j)$; therefore $j \in A_i(r)$. If $j \in A_i(r)$, then $\mathrm{d}(i,j) \leq r$ and j was active after round $r - \mathrm{d}(i,j)$. This implies that j's BFS message arrives at i not later than round r; therefore, $j \in I_i(r)$. □

Lemma 3.2. *If $c_i(r+2) \geq 2$, then $A_i(r) = V$.*

Proof. If $A_i(r) \neq V$ then there are two nodes, $u \in A_i(r)$ and $v \notin A_i(r)$ which are adjacent, i.e., $\mathrm{d}(u,v) = 1$. This means that $s_u(r - \mathrm{d}(i,u)) =$ ACTIVE and $s_v(r - \mathrm{d}(i,v)) =$ QUIESCENT. Since u and v are adjacent, then $s_v(r + 1 - d(i,u)) =$ ACTIVE. There are three possible cases: (1) $\mathrm{d}(i,u) = \mathrm{d}(i,v)$, in which case v became active in round $r+1-\mathrm{d}(i,v)$, and i receives the BFS from v at round $r+1$; (2) $\mathrm{d}(i,v) = \mathrm{d}(i,u)+1$, in which case $s_v(r+2-\mathrm{d}(i,v)) =$ ACTIVE, and i receives the BFS from v at either round $r+1$ or round $r+2$; (3) $\mathrm{d}(i,v) = \mathrm{d}(i,u) - 1$, cannot happen, as it contradicts $s_v(r - \mathrm{d}(i,v)) =$ QUIESCENT. In any case, $c_i(r+1) = 0$ or $c_i(r+2) = 0$. Therefore, since c_i can only increase 1 unit per round, it follows that if $c_i(r+2) \geq 2$, then $A_i(r) = V$. □

Lemma 3.3. *After any round r, $e_i(r) = \max(\{0\} \cup \{\mathrm{d}(i,j) \mid j \in I_i(r)\})$.*

Proof. Trivial induction on the number of rounds. □

Theorem 3.1 (eccentricity convergence)**.** *If $c_i(r+2) \geq 2$, then $e_i(r) = \mathrm{ecc}(i)$.*

Proof. Combine the previous three lemmas. □

Lemma 3.4. *If for some r and $n \geq 2$, $c_i(r+n) = n$, then for all k, $c_i(r+k) = k$.*

Proof. After the first round r such that $c_i(r+2) = 2$, by the first two lemmas $I_i(r) = V$. This implies that for $r' \geq r$, $c_i(r'+1) = c_i(r') + 1$. □

Theorem 3.2 (diameter convergence)**.** *When $c_i(r) \geq 2$ and $c_i(r) > d_i(r)$, then $d_i(r) = D$.*

Proof. By contradiction. Assume $c_i(r) \geq 2$ and $c_i(r) > d_i(r)$ but $d_i(r) < D$. By Theorem 3.1, $e_i(r) = \mathrm{ecc}(i)$. Also, it is trivial that $d_i(r) \geq e_i(r)$. Then, as in a graph all the eccentricities between the radius and the diameter are present [3], there exists two nodes u and v with $\mathrm{d}(u,v) = d_i(r) + 1$. Assume without loss of generality that $\mathrm{d}(i,u) \geq \mathrm{d}(i,v)$. From the previous lemma, for $r' = r - c_i(r)$, it follows that $c_i(r'+k) = k$. As $c_i(r'+2) = 2$, by Lemma 3.2, $A_i(r') = V$; therefore $s_u(r' - \mathrm{d}(i,u)) =$ ACTIVE. Then, $d_v(r' - \mathrm{d}(i,u) + \mathrm{d}(u,v)) \geq \mathrm{d}(u,v)$ (BFS from u has reached v). Furthermore, $d_i(r' - \mathrm{d}(i,u) + \mathrm{d}(u,v) + \mathrm{d}(i,v)) \geq \mathrm{d}(u,v)$ (DIAM message from v has reached i). Since $\mathrm{d}(i,u) \geq \mathrm{d}(i,v)$, then $d_i(r' + \mathrm{d}(u,v)) \geq \mathrm{d}(u,v)$. Recall that $\mathrm{d}(u,v) = d_i(r) + 1$, and let $r'' = r' + \mathrm{d}(u,v) = r - c_i(r) + d_i(r) + 1$. From the assumption $c_i(r) > d_i(r)$, it means that $r'' \leq r$, which together with $d_i(r'') \geq d_i(r) + 1$ contradicts the monotonicity of d_i. □

Theorem 3.3 (radius convergence)**.** *When $c_i(r) \geq 2r_i(r)$, then $r_i(r) = R$.*

Proof. We assume networks with at least one link and two nodes, which means $r_i(r) \geq 1$. If $c_i(r) \geq 2r_i(r)$, we have $c_i(r) \geq 2$, which means that, from Lemma 3.4, all BFSs have already reached node i after round $r' = r - c_i(r)$, and from r' on we have $c_i(r'+k) = k$. Assume, by way of contradiction, that $r_i(r) > R$. Then, at most after round $r' + r_i(r) - 1$, all BFSs have reached some node u in the center of the network. At most two rounds later, after round $r'' = r' + r_i(r) + 1$, we have $c_u(r'') \geq 2$ and u has sent a RAD message with the network radius. At most $r_i(r) - 1$ rounds later, after round $r''' = r' + 2r_i(r)$ this message arrives at i and $r_i(r''') = R$. But assuming $c_i(r) \geq 2r_i(r)$ it means that $r''' \leq r$, which contradicts r_i being monotonically decreasing. As r_i results from some eccentricity and is always an upper bound of the radius, we must have $r_i(r) = R$. □

3.3 Convergence Bounds

We now determine upper bounds on the number of rounds for convergence of eccentricity, diameter and radius. Given that we have described the algorithm for the general case of variable starting times, what matters is the number of rounds after the first activation; i.e., ignoring an initial sequence of rounds with all nodes inactive. Therefore, in this section we consider that the first node became active after round 0; round 1 is when the first non-environment message is sent.

Proposition 3.1 (eccentricity bound)**.** *Node i can determine its eccentricity at most in $D + \mathrm{ecc}(i) + 2$ rounds.*

Proof. After round D all nodes are active, so the last BFS arrives at i at most after round $D+\mathrm{ecc}(i)$. Two rounds later the c_i variable reaches 2 and from Lemma 3.1 the eccentricity has already converged. □

Proposition 3.2 (diameter bound)**.** *Node i can determine the network diameter at most in $2D + \mathrm{ecc}(i) + 1$ rounds.*

Proof. After round D all nodes are active, so the last BFS arrives at i at most after round $D+\mathrm{ecc}(i)$. Subsequently c_i starts increasing and after further $D+1$ rounds the local condition $c_i > d_i$ is met. As we are considering networks with at least one link, i.e., $D \geq 1$, then at this round we have also $c_i \geq 2$ and from Theorem 3.2 the diameter has converged.

□

Proposition 3.3 (radius bound)**.** *Node i can determine the network radius at most in $D+\mathrm{ecc}(i)+2R$ rounds.*

Proof. After round D all nodes are active, so the last BFS arrives at i at most after round $D+\mathrm{ecc}(i)$; afterwards c_i starts increasing and after round $r = D+\mathrm{ecc}(i)+2R$ we have $c_i(r) \geq 2R$. Also, after at most round $D+R$ all BFS have arrived at all center nodes; two rounds later, at most after round $r' = D + R + 2$, each center node sends a RAD message containing R. There are two possibilities: (1) $\mathrm{ecc}(i) > 1$, the RAD message from a center node j arrives at i at most R rounds later, which means that at most after round $r'' = D + 2R + 2$ we have $r_i(r'') = R$; given that $r'' \leq r$, then $r_i(r) = R$, and the local radius convergence criteria $c_i(r) \geq 2r_i(r)$ is met; (2) $\mathrm{ecc}(i) = 1$, in which case $R = 1$, i is a center node, and at round r' we have $r_i(r') = R$; as in this case $r' = r$, we have $c_i(r) \geq 2r_i(r)$ as well. □

Corollary 3.1. *All nodes know: their eccentricity at most in $2D+2$ rounds; the diameter at most in $3D + 1$ rounds; and the radius at most in $2D + 2R$ rounds.*

3.4 Termination

To keep the presentation clear and avoid cluttering, we did not include in the algorithm the mechanics of termination; i.e., each node reaching a "terminated" state in which it stops sending messages. In general distributed termination is independent from reaching some result, and nodes may have to keep propagating messages for some time.

In this case, however, it is easy to see that when a node has determined both the radius and diameter through the local convergence criteria, all neighbors will have the same criteria met after at most one more round. (After one more round, each node j neighbor from i, will have c_j with at least the same value node c_i had, and both r_j and d_j will have the same values as in node i.) Therefore, after having met both criteria for radius and diameter, a node needs only execute one more round and stop.

3.5 Improving Storage Requirements

In the previously described algorithm each node accumulates in the I variable all ids received in all previous rounds. Although it has made the description intuitive and streamlined proofs, it means that, regardless of network topology, by the end of the execution each node will need to store $\Theta(|V|)$ ids.

Here we show that it is enough to keep in the state only the ids received in the two previous rounds. While this modification does not change the worst case space requirement complexity (it still remains $O(|V|)$ ids), it may be useful in practice. As an example, for 2D geometrical networks (e.g. a geographically spread sensor network with links according to inter-node distance), under synchronized start times, the number of ids that arrive in a single round will be $O(\sqrt{|V|})$.

The modification to the algorithm is trivial and consists of replacing state variable I by a pair I, J used as a sliding window; replacing the test $j \notin I_i$ with $j \notin I_i \cup J_i$ and replacing $I'_i = I_i \cup \{j \mid \langle \text{BFS}, j, \_\rangle \in M''\}$ with $I'_i = J_i$ and $J'_i = \{j \mid \langle \text{BFS}, j, \_\rangle \in M''\}$. The modification is possible due to the following property of the original algorithm.

Proposition 3.4. *A node j can only receive* BFS *messages $\langle \text{BFS}, i, \_\rangle$, originated in a node i activated at round r, in rounds $r + \mathrm{d}(i,j)$, $r + \mathrm{d}(i,j) + 1$ and $r + \mathrm{d}(i,j) + 2$.*

Proof. The first round, where node j can receive a $\langle \text{BFS}, i, \_\rangle$ message, is $r' = r + \mathrm{d}(i,j)$, by the shortest path from i; j rebroadcasts it and at round $r' + 1$ it arrives at all neighbors, if any. Also at round $r' + 1$ node j may receive such a message if there exists a neighbor node u at the same distance from i (i.e., $\mathrm{d}(i,u) = \mathrm{d}(i,j)$); this neighbor, similarly to j, has received such a message at round r' and has rebroadcasted it. At round $r' + 2$ node j can receive such a message if it has a neighbor u one hop further away from i (i.e., $\mathrm{d}(i,u) = \mathrm{d}(i,j) + 1$); this neighbor has received the message at round $r' + 1$ and has rebroadcasted it at round $r' + 2$. Because any neighbor u of j has stored i in the respective I_u variable after at most round $r' + 1$, it will not rebroadcast any $\langle \text{BFS}, i, \_\rangle$ message in any round later than $r' + 2$, from which such messages cannot reach j later than round $r' + 2$. □

4 Related Work

As mentioned in the introduction, our algorithm improves the classic modular approach [8], where BFS trees are first computed at each node and later reused to perform a global computation of the network radius and diameter (in at most $4D + 2$ rounds). By Corollary 3.1, we achieve a speedup of D rounds for computing the diameter, and since $D \leq 2R \leq 2D$ the speedup for computing the radius varies between 2 and $D + 2$ rounds. The maximum speedup occurs, for example, in path graphs. This improvement is achieved with the same message complexity of $\Theta(|V|\,|E| \log |V|)$ bits, and the same space complexity of $O(|V|)$ ids and computation complexity per round of $O(|V|^2)$ at each node.

The work in [10] computes the diameter under the more restrictive synchronized start time model, where all nodes are activated at the first round. This is a fast algorithm since they also disseminate candidate values for eccentricities before they converge. However, since they assume that all nodes are active in the first round the local termination criteria is much simpler. When restricted to a setting where all nodes are active in the first round our algorithm outputs the diameter in the same time bound. Even though we are more general, we have significant improvements in message and space complexity.

Related to the computation of the radius and the eccentricity is finding the network center. To the best of our knowledge, within similar bounds for space and processing complexity per round, the fastest algorithm so far to find network center nodes was proposed by Korach, Rotem and Santoro [6]. This algorithm builds on a simpler algorithm to find a center of a tree, and the observation that the center of a general network must also be the center of its own BFS tree: the initiator node triggers BFS generation at all nodes, picks a center among all candidates which are centers of their own BFS, and later disseminates this information to all nodes. The closest

center node c to the initiator node i will receive the confirmation that it is indeed a center of the network by round $4R + \mathrm{d}(i,c) + 1$, with total message complexity of $\Theta(|V|\,|E| \log |V|)$ bits. Notice that, in contrast to this, our algorithm is fully symmetrical, not requiring a distinguished node as initiator (which would have to be chosen in some way, e.g. by a leader election). Even discounting this factor, our more general algorithm has no time penalty: in fact, it can be shown that it even improves this upper bound by at least one round.

A more general problem than computing the eccentricities, radius and diameter is that of computing the distance matrix or all-pairs shortest path matrix in a network. In fact, relying on this connection, a faster distributed variant of our algorithm could be designed to compute the radius and diameter: instead of propagating just the distances, each node propagates sets of neighbors – at most by round $2D + 2$ all nodes would be able to determine the full topology of the network and run a standard unweighted all-pairs shortest path algorithm with computation complexity $O(|V|\,|E|)$ at the last round. Unfortunately, even assuming that computation complexity per round is negligible, this algorithm requires $\Theta(|V|^2)$ space complexity at each node, which is impractical for large networks. Moreover the message complexity increases to $\Theta(|E|^2 \log |V|)$ bits, and requires a bandwidth of $O(|E| \log |V|)$ bits per link.

With such large bandwidth it is possible to decrease the message complexity using more elaborated approaches. For example, Kanchi and Vineyard [5] propose a distributed algorithm to compute the all-pairs shortest path matrix with message complexity of $O(|V|\,|E| \log |V|)$ bits: a spanning tree is first computed using the algorithm proposed by Awerbuch [2] and later reused to propagate the topological information to a root node that computes the distance matrix and disseminates it to all nodes. The tradeoff for this optimization is, unfortunately, a substantial increase in the number of rounds, although still $O(|V|)$.

Message complexity can also be reduced by trading off accuracy. For example, Gu and Cheng [4] propose a distributed algorithm to compute an estimate $\hat{D}$ for the diameter, such that $D \leq \hat{D} \leq D + 2$. Unfortunately, the bandwidth requirements are slightly worse than ours and the improvement in message complexity does not extend to time complexity: although the authors do not quantify this measure, it is clear from the algorithm presentation that the number of rounds until completion is substantially larger than ours. Moreover, this algorithm also requires a distinguished node as initiator.

5 Conclusions

In this paper we propose a time efficient algorithm that computes in all nodes of a network the values of the node eccentricity, and the network diameter and radius. The algorithm is very flexible in the sense that it does not require a distinguished node, one or more nodes can initiate the computation, and concurrent initiations have no detrimental impacts on the various bounds and complexities.

Under the same communication, space, and computation complexity, the presented algorithm significantly improves existing time bounds for the diameter and radius computation. It also slightly improves the time bounds for the special case of finding center nodes, while relaxing the need for a special initiator.

The key to the improvement was to abandon the traditional modular approach. Instead, our algorithm relies on a very early propagation and aggregation of the maximum and minimum candidate eccentricities, even before these values have stabilized. Together with adequate convergence detection criteria, this allowed a simple and fast approach to the computation of these distances.

References

[1] Baruch Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.

[2] Baruch Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems (detailed summary). In *STOC*, pages 230–240. ACM, 1987.

[3] Fred Buckley and Frank Harary. *Distance in Graphs*. Addison-Wesley, 1990.

[4] Qian-Ping Gu and Zixue Cheng. Efficient estimation of diameter for distributed networks. In *Proc. of the 11th Annual International Symposium on High Performance Computing Systems*, pages 261–268, 1997.

[5] Saroja Kanchi and David Vineyard. An optimal distributed algorithm for all-pairs shortest-path. *Information Theories and Applications*, 11(2):141–146, 2004.

[6] Ephraim Korach, Doron Rotem, and Nicola Santoro. Distributed algorithms for finding centers and medians in networks. *ACM Trans. Program. Lang. Syst.*, 6(3):380–401, 1984.

[7] Sung-Ju Lee, Elizabeth M. Belding-Royer, and Charles E. Perkins. Scalability study of the ad hoc on-demand distance vector routing protocol. *Int. Journal of Network Management*, 13(2):97–114, 2003.

[8] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[9] David Peleg. Time-optimal leader election in general networks. *J. Parallel Distrib. Comput.*, 8(1):96–99, 1990.

[10] Boleslaw K. Szymanski, Yuan Shi, and Noah S. Prywes. Terminating iterative solution of simultaneous equations in distributed message passing systems. In *PODC*, pages 287–292, 1985.