The Power of a Leader in the Stone Age

Yuval Emek*
yemek@ie.technion.ac.il
Technion

Stephan Holzer†
holzer@mit.edu
MIT

Roger Wattenhofer
wattenhofer@ethz.ch
ETH Zurich

Abstract

It is known that in cellular networks modeled by the *stone age model of distributed computing* [PODC 2013 [3]] certain computations are not possible. This includes electing a leader, computing shortest paths and the diameter of a graph [STOC 1980 [1]],[ICALP 2014 [2]]. However, certain organisms such as the true slime mold *Physarum polycephalum* seem to be able to compute shortest paths [Nature 2000 [4]] and structures close to minimum spanning trees [Science 2010 [6]]. When such an organism is modeled using the stone age model, this behavior is only possible when a leader is available. In a first attempt to explain the true slime molds abilities using the stone age model we equip the cellular network with a leader and show that this enables the network to identify cells of maximal distance to each other.

1 Introduction

Experiments indicate that the true slime mold might be able to compute shortest paths and graphs close to minimum spanning trees over food sources in space [4, 6]. Although the true slime mold consists of only one (giant) cell or merged cells, it has multiple nuclei [5]. When we assume that these nuclei communicate/interact with each other through tubular structures [5], the true slime mold is a candidate to be modeled by the stone age model of distributed computing [3] which essentially corresponds to networked finite state machines. However, it is known that computations such as shortest paths cannot be performed in the stone age model. In particular it was shown [1, 2] that when we consider Las Vegas algorithms in a version of the local model, leader election and solving many related problems is impossible. This extends to the stone age model which has further restrictions compared to the model studied in [1, 2]. Thus the construction [1, 2] implies that in the stone age model it is impossible to identify two nodes of maximal distance in the network, or to compute shortest paths between two nodes as one can even not distinguish a graph with diameter 1 (the ring of size 3) from a graph of diameter $2n$ (a ring of size $4n$).

In this paper we show that when a leader is given in addition to the assumptions made in the stone age model, the network can identify two nodes of maximal distance to each other in the graph. This essentially corresponds to computing the diameter in this model, as each finite state machine cannot store a number representing the diameter. By doing so we open the doors for further investigations of this model when a leader is provided. This might eventually lead to algorithms that identify which nodes/nuclei are included in a shortest paths connecting two food sources. More sophisticated problems such as determining which nuclei are contained in the paths

*Part of this work has been done at ETH Zurich.

†Part of this work has been done at ETH Zurich. At MIT the author is supported by the following grants: AFOSR Contract Number FA9550-13-1-0042, NSF Award 0939370-CCF, NSF Award CCF-1217506, NSF Award number CCF-AF-0937274.

used as edges between food sources in a minimum spanning tree (over the food sources) might be possible as well or require a further modification of the model. The goal of this paper is to provide a first step towards a more general study with the aim to understand weather the property of having a leader (or being able to elect a leader) is a property that determines which (global) problems can be solved. Here, global problems are those that require nodes at distance $\Omega(D)$ to communicate with each other in order to compute a solution, where D is the diameter of the graph. Another open question is whether there are any local problems in the stone age model that cannot be solved without a leader but can be solved when we know a leader?

2 Model and Definitions

We recall the stone age model of distributed computing defined in [3] using their notation in an informal. The formalism of [3] is omitting as this paper only provides high-level implementations of the algorithms that do not use specific details of the model [3]. We assume a network represented by a finite undirected graph $G = (V, E)$. Under the *networked finite state machines (nFSM)* model, each node $v \in V$ runs a protocol Π executed by a finite state machine. It is important to point out that protocol Π is oblivious to the graph G.

Communication. Node v communicates with its adjacent nodes in G by *transmitting* messages. A transmitted message consists of a single letter σ from an alphabet Σ and it is assumed that this letter is delivered to all neighbors u of v. We assume that any of the constant many messages sent during our protocol correspond to a letter in Σ. Each neighbor u has a *port* $\psi_u(v)$ (a different port for every adjacent node v) in which the last message σ received from v is stored.

Asynchrony. The nodes are assumed to operate in an *asynchronous* environment as described in [3]. In the same paper [3] it is shown how to implement a synchronous environment. Therefore we only consider synchronous executions of our algorithms and assume time is partitioned into time slots, where each node can send one message per time slot.

In this paper we are interested in possibility results and do not introduce the notion of runtime. If one would consider the runtime, the presented algorithm would have a runtime close to the worst possible runtime in this model and it is an open problem to improve it.

Definition 2.1 (Diameter)**.** *Let $G = (V, E)$ be a graph and let $u, v \in V$ be any two nodes in G. The distance $d(u, v)$ between $u$and v is the length of a shortest path between u and v. The diameter $D := \max_{u,v \in V} d(u, v)$ of a graph G is the maximum distance between any two nodes of the graph.*

3 Identifying two Nodes of Maximal Distance

Theorem 3.1. *A network $G = (V, E)$ modeled by the stone age model of computation [3] extended by the existence of a leader is able to determine two nodes $u, v \in V$ of maximal distance in the network.*

Proof. In this short paper we present only a high-level description of the algorithm in the stone age model. We start by considering the following algorithm that computes the diameter in classical models of computation and is stated in a way that is useful for investigation in the stone age model:

Algorithm Compute the diameter. In our setting this corresponds to finding two nodes of maximal distance.

```
1:  u_max, v_max are any two nodes in V;
2:  for u ∈ V do
3:     for v ∈ V do
4:        if d(u, v) > d(u_max, v_max) then
5:           u_max := u;
6:           v_max := v;
7:        end if
8:     end for
9:  end for
10: output u_max, v_max;
```

We implement the above algorithm in the stone age model when a leader is provided.

Line 1: Given a leader l, we set $u_{\max}, v_{\max} := l$ in the first line.

Line 2: As a first step towards iterating over all nodes, we argue that given a leader we can choose a random node $u' \in V$. In this sub-procedure we require that each node has a positive probability to be chosen. We can do so by starting a random walk at the leader l. That is l chooses a random neighbor w. With probability $1/2$, node w decides to be the randomly selected node. If not, node w chooses a random neighbor w'. With probability $1/2$, node w' decides to be the randomly selected node and so on.

Now we use this insight to iterate over all $u \in V$: Start by choosing a random $u' \in V$. Node u' marks itself and the inner part of the for-loop is executed. We select another node $u' \in V$ at random. If u' is unmarked, it marks itself and we execute the inner part of the for-loop. Continuing like this until all nodes are marked ensures that we will have iterated over all nodes $u \in V$ in the end. All this can be done by a randomized FSM.

We still need to ensure that we can detect when all nodes are marked. After each iteration the leader l does the following: l sends out a message "Are you not marked?" that is broadcast through the network and waits for some time to be specified below. If a node receives this message and is not marked, it broadcasts "No, I am not marked" through the network. Parallel to this, when node $u_{\max}$ receives the message "Are you not marked?" for the first time during this iteration, $u_{\max}$ broadcasts a message "Ping" that is received by $v_{\max}$ after some time. This causes $v_{\max}$ to broadcast "Ping 2", which in turn is received by $u_{\max}$ after some time. Then $u_{\max}$ broadcasts "Stop".

We claim that if l does not receive message "No, I am not marked" before receiving "Stop", then all nodes are marked: Since $u_{\max}$ and $v_{\max}$ are temporarily the nodes of maximal distance that the algorithm is aware of, the leader would have received the message "No, I am not marked" within $2 \cdot d(u_{\max}, v_{\max})$ time slots if there was any. On the other hand it takes at least $2 \cdot d(u_{\max}, v_{\max})$ time steps before $u_{\max}$ broadcasts "Stop". Again, all this can be executed by an FSM.

Line 3: We do the same as in Line 2 but use a different alphabet during the computation.

Line 4: The high-level idea is that u and $u_{\max}$ start a broadcast at the same time. If $d(u, v) > d(u_{\max}, v_{\max})$, then $u_{\max}$'s broadcast reaches $v_{\max}$ earlier than u's broadcast reaches v and is able to detect this.

Since u and $u_{\max}$ do not have a clock and do not know how to start at the same time, we simulate synchronized time steps using the leader. The leader sends a message and waits until it receives a reply from both u and $u_{\max}$. Then one simulated time step is over. Upon receiving the message belonging to this time step, all nodes active in each of the two ongoing broadcasts execute one step. When these broadcasts reach v or $v_{\max}$, these nodes broadcast "v reached", or "$v_{\max}$ reached". Based on this the leader decides whether $d(u, v) > d(u_{\max}, v_{\max})$. This can be executed by an FSM.

Lines 5,6: The two nodes rename themselves.

Line 10: After all nodes in the first for-loop are marked, the nodes that are currently $u_{\max}$ and $v_{\max}$ will have maximal possible distance in G.

Finally we want to note that there are only 4 variables involved such that each node knows which role it plays during our implementation. Also it is not be the case that a node has to broadcast the same message two times after each other due to two parallel (or sequential) broadcasts using the same message. In between two broadcasts that use the same message there is always another broadcast using a different message. Thus, during a broadcast, a node only needs to send the received message if it differs from the last message sent.

If there are several different broadcasts going on in parallel, they can be simulated by extending the alphabet such that there is no congestion. This is important e.g. when we need to decide $d(u, v) > d(u_{\max}, v_{\max})$. □

References

[1] Dana Angluin. Local and global properties in networks of processors. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 82–93. ACM, 1980.

[2] Yuval Emek, Jochen Seidel, and Roger Wattenhofer. Computability in anonymous networks: Revocable vs. irrecovable outputs. In *Automata, Languages, and Programming (ICALP)*, pages 183–195. Springer, 2014.

[3] Yuval Emek and Roger Wattenhofer. Stone age distributed computing. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing (PODC)*, pages 137–146. ACM, 2013.

[4] Toshiyuki Nakagaki, Hiroyasu Yamada, and Ágota Tóth. Intelligence: Maze-solving by an amoeboid organism. *Nature*, 407(6803):470–470, 2000.

[5] Atsuko Takamatsu, Eri Takaba, and Ginjiro Takizawa. Environment-dependent morphology in plasmodium of true slime mold¡ i¿ physarum polycephalum¡/i¿ and a network growth model. *Journal of theoretical biology*, 256(1):29–44, 2009.

[6] Atsushi Tero, Seiji Takagi, Tetsu Saigusa, Kentaro Ito, Dan P Bebber, Mark D Fricker, Kenji Yumiki, Ryo Kobayashi, and Toshiyuki Nakagaki. Rules for biologically inspired adaptive network design. *Science*, 327(5964):439–442, 2010.