CLIQUE HERE:
ON THE DISTRIBUTED COMPLEXITY IN FULLY-CONNECTED NETWORKS

BENNY APPLEBAUM[1]
School of Electrical Engineering
Tel Aviv University
Israel

DARIUSZ R. KOWALSKI
Department of Computer Science
University of Liverpool
UK

BOAZ PATT-SHAMIR[2,3]
School of Electrical Engineering
Tel Aviv University
Israel

ADI ROSÉN[3,4]
CNRS and Université Paris Diderot
France

Received: June 2014
Revised: October 2014
Communicated by D. Peleg

ABSTRACT

We consider a message passing model with n nodes, each connected to all other nodes by a link that can deliver a message of B bits in a time unit (typically, $B = O(\log n)$). We assume that each node has an input of size L bits (typically, $L = O(n \log n)$) and the nodes cooperate in order to compute some function (i.e., perform a distributed task). We are interested in the number of rounds required to compute the function.

We give two results regarding this model. First, we show that most boolean functions require $\lceil L/B \rceil - 1$ rounds to compute deterministically, and that even if we consider randomized protocols that are allowed to err, the expected running time remains $\Omega(L/B)$ for most boolean function. Second, trying to find explicit functions that require super-constant time, we consider the *pointer chasing* problem. In this problem, each node i is given an array A_i of length n whose entries are in $[n]$, and the task is to find, for any $j \in [n]$, the value of $A_{n-1}[A_{n-2}[\ldots A_0[j]\ldots]]$. We give a deterministic $O(\log n / \log\log n)$ round protocol for this function using message size $B = O(\log n)$, a slight but non-trivial improvement over the $O(\log n)$ bound provided by standard "pointer doubling." The question of an explicit function (or functionality) that requires super constant number of rounds in this setting remains, however, open.

Keywords: CONGEST Model, Communication Complexity, Pointer Jumping, Network Algorithms

[1]Supported by ISF grant 1155/11, Israel Ministry of Science and Technology (grant 3-9094), GIF grant 1152/2011, and by the Check Point Institute for Information Security.

[2]Supported in part by the Israel Science Foundation (grant No. 1444/14).

[3]Supported in part by a grant from the Israel Ministry of Science, Technology and Space, Israel and the French Ministry of Higher Education and Research (Maimonide 31768XL).

[4]Supported in part by ANR project RDAM (ANR-12-BS02-005).

1 Introduction

In the LOCAL model of network algorithms, each link can carry a message of arbitrary length in one time unit, effectively assuming that link bandwidth is unbounded [14]. This model is therefore allows us to study the *distance* that input items must travel in order to produce correct output. In the present paper we consider a model that trivializes the distance constraints but restricts *bandwidth* by considering a clique and putting a cap on the message size, thus allowing us to isolate the effect of congestion on the complexity of distributed computation.

More specifically, we assume that the network topology is an n-node clique whose links can carry B bits in a time step. B is a given parameter, and typically it is assumed that $B = \Theta(\log n)$. In this special case of the CONGEST model [14], any B-bits input item can be retrieved by any node in a single time unit, but if the number of inputs is much larger than n, then the trivial centralized solution of collecting all inputs at a single node is impossible. We are interested in the time complexity (number of rounds) of computing various functions and functionalities in this model. From a (multiparty) communication complexity point of view, our setting is the number-in-the-hand model (see [7]), where each link can carry B bits per time unit, and the objective is to minimize the number of rounds required to compute the function (as opposed to the more common question of minimizing the total number of bits transmitted). We note that in this model (as a special case of the CONGEST model, and using a streamlined version of the α synchronizer [2]) we may assume that the network is synchronous, i.e., computations proceed in global rounds.

We assume that the input size at each node is L bits, where L is a given parameter. We focus on the case where $L = \Theta(nB)$. This allows us to consider inputs which describe graphs using $O(B)$ bits per edge. The main question that drives us is whether there are problems that require super-constant number of rounds in this model. The simple lower bound argument that "all input must be communicated" does not hold, because the complete input can be communicated in this case in a constant number of rounds. No such decision problems are known to date, and on the other hand some important functionalities can be computed in constant time, e.g., selection and sorting [13, 10, 8].

Our results. Our contribution in this paper is twofold. First, we give an existential proof that shows that there are predicates (functions with a single bit of output) that require super-constant number of rounds. In fact, we show that "most" predicates require $\lceil L/B \rceil - 1$ rounds to compute deterministically (Section 3). Moreover, even if we consider randomized protocols that are allowed to err, the *expected* number of rounds remains $\Omega(L/B)$ for "most" predicates. In our quest to find an explicit function (or functionality) that requires a super-constant number of rounds, we consider the *pointer chasing* problem. In this problem, each node i is given an array A_i of length n whose entries are in $[n]$, and the task is to find, for any $j \in [n]$, the value of $A_{n-1}[A_{n-2}[\ldots A_0[j]\ldots]]$. This function seems to be a natural candidate for a lower bound of the type we look for because of its apparent sequential specification. Moreover, this function has been extensively used in the communication complexity literature [12, 15, 3, 6]. It is not hard to see that using the "pointer doubling" technique, this problem can be solved in $O(\log n)$ rounds in our setting (here $L = O(n \log n)$, and assuming $B = O(\log n)$). While we fall short of our original goal of proving a super-constant lower bound for pointer chasing, we make a slight, but non-trivial, improvement on the upper bound side and give an $O(\log n / \log\log n)$-round deterministic protocol (Section 4). The question of finding an explicit

function (or functionality) with a super constant lower bound in this setting remains, however, open.

1.1 Related work

The Clique model. The special case of the fully-connected network (clique) in the CONGEST model was first considered by Lotker *et al.* [11]. For simplicity, we state all results below under the assumption that $B = \Theta(\log n)$ and $L = O(n \log n)$. In [11] it is shown that if each link has a weight known to its endpoints, then a minimum-weight spanning tree can be found in $O(\log \log n)$ rounds. The next work in this model [13] gives a randomized algorithm to sort n^2 items, where each player receives as input n items. The goal is that each player knows the rank of each of its input items, among the global n^2-items input. The running time of the algorithm in [13], using a result of Lenzen and Wattenhofer [10], is constant in expectation and w.h.p. Lenzen subsequently improved the result by showing that sorting can be solved *deterministically* in constant time [8]. In [8] it is also shown that any routing instance, in which each node is the source of at most n messages and at most n messages are destined at each node, can be performed in $O(1)$ deterministic time. We note that the latter result allows one to consider algorithms and lower bounds designed for the BSP model [17], where it is assumed that in every "round," each node can send and receive at most h messages for a given parameter h of the model. In the clique model, $h = n$.

We note that Lenzen and Wattenhofer [9] considered a similar but "globally restricted" model, where in each communication round, $O(n^\epsilon)$ bits can be transferred (possibly all from a single source to a single destination). They show that any algorithm that uses only n^ϵ bits of memory per node, over networks of polylogarithmic maximal degree, and runs in T rounds in the LOCAL model, can be emulated in the globally restricted model in time $O(\log T + T \log \log n / \log n)$, which means that any problem with a log-time algorithm in the polylog-bounded-degree CONGEST model has an $O(\log \log n)$ time algorithm in the globally-restricted model.

Finally, we point out that concurrently with our work, Drucker *et al.* [5] have reported an existential lower bound of $(n - 5 \log n)/B$ rounds for the deterministic computation of a boolean function in the case where $L = n$. They also prove a transformation of a certain class of circuits to the clique model, therefore showing that a lower bound on the clique model implies a lower bound on that class of circuits. Also, in [5] the "Clique-Bcast" model is considered, where in each round each node can broadcast a single B-bit message that is received by all.

Pointer Chasing. The problem of pointer chasing is an important problem in communication complexity theory, where it was studied mainly in the 2-party setting. In this setting, it is assumed that Alice receives an array A of n entries, each in the range $[n]$, and similarly Bob receives an array B. Given k, the goal is to compute $A[B[A \ldots A[1] \ldots]]$ where dereferencing is done k times starting with Alice (e.g., for $k = 2$ the task is to find $B[A[1]]$). While it is easy to compute the output in k rounds starting with A, using a total of $O(k \log n)$ bits of communication, it was shown that if only $k - 1$ communication rounds are available (or when k rounds are available, but the first to speak is Bob), then the communication complexity jumps to $\Theta(n \log^{(k-1)} n)$ for all fixed k, where $\log^{(k)}$ is the k-iterated logarithm function [12, 4, 15]. Another version of pointer chasing, where the input to the two players represents an n-ary tree, rather than a bipartite graph of $2n$ nodes, is used in [6] to derive lower bounds on the communication complexity over noisy channels. The multiparty communication

complexity literature has considered the problem in our formulation (i.e., the input represents a layered graph) but in the number-on-the-forehead model (cf. [7]) which is different than our setting (see, e.g., [4, 3]).

The pointer doubling technique is due to Shiloach and Vishkin, who invented it in the context of computing connected components in the PRAM model [16].

2 The Model

The model we consider consists of n players $\mathcal{P}_i$, $0 \leq i \leq n-1$, where each pair of players is connected by a bidirectional communication link (i.e, the communication graph is an undirected clique). The system is synchronous, and in each time step, each communication link can transmit B bits, in each direction, for some given integer parameter $B > 0$. We note that it is common to assume that $B = \Theta(\log n)$, in the spirit of the word model in classical RAM (see, e.g., [1]).

Each player $\mathcal{P}_i$, $0 \leq i \leq n-1$, receives as input a string of L bits denoted x_i, for some given integer parameter $L > 0$. We assume that $x_i \in \mathcal{S}$, for some $\mathcal{S} \subseteq \{0,1\}^L$. The task of the players is to compute some functionally $f(x_0, \ldots, x_{n-1}) \to (y_0, \ldots, y_{n-1})$, such that each player $\mathcal{P}_i$, $0 \leq i \leq n-1$, outputs a bit string $y_i \in \mathcal{T} \subseteq \{0,1\}^M$ for some given integer parameter $M \geq 0$. We shall consider the case of $L > B$; otherwise, all inputs can be gathered by each single player in a single round. Note that in many cases it is desirable that the input describes a graph over the players, in which case typically $L = \Theta(n \log n)$, because this way we can attach to each possible incident edge $O(\log n)$ bits (for example, specifying the edge weight). We assume that each player $\mathcal{P}_i$ knows its index i in the sense that it can be used by the code of the program the player runs.

A protocol P runs in rounds, where in each round each player sends to each other player a message of B bits, on the link connecting them. At any round, in addition to sending messages, each player can write its output. This is formalized by assuming a special write-once output tape.

Given a function f, we say that a run of a protocol P is *correct* if for the vector of inputs given to the players, $\vec{x}$, and for the outputs written by the players, $\vec{y}$, it holds that $f(\vec{x}) = \vec{y}$.

The running time of a deterministic protocol P on a given input $\vec{x}$, denoted $T_P(\vec{x})$, is the number of rounds until all players have written on their output tape.

Definition 1. Given a function f, $t \geq 1$, and $0 \leq p \leq 1$, a deterministic protocol P is said to $D(t,p)$-*solve* f if (1) for any $\vec{x} \in \mathcal{S}^n$, $T_P(\vec{x}) \leq t$; and (2) there is a set $\mathcal{X} \subseteq \mathcal{S}^n$, $|\mathcal{X}| \geq p \cdot |\mathcal{S}^n|$, and the protocol is correct when run on any $\vec{x} \in \mathcal{X}$.

Definition 2. Given a function f, $t \geq 1$, and $0 \leq p \leq 1$, a randomized protocol P is said to $R(t,p)$-*solve* f if (1) for any $\vec{x} \in \mathcal{S}^n$, $E[T_P(\vec{x})] \leq t$; and (2) for any $\vec{x} \in \mathcal{S}^n$, the probability that the protocol is correct when run on $\vec{x}$ is at least p. The probabilities (and expectations) are over the random choices of the protocol.

We note that since the bandwidth of the communication links is bounded there may be a difference in the round complexity of randomized protocols if we allow shared randomness or not. For the lower

bound we present we allow the parties to have shared randomness (which makes the lower bound only stronger). Our upper bound is deterministic so it applies in both models.

3 Lower Bound

In this section we prove that in the clique model, "most" boolean functions cannot be computed deterministically in less than $\lceil L/B \rceil - 1$ rounds.[1] Note that in the clique model, any function can be computed in $\lceil L/B \rceil$ rounds, simply by sending all input to all players, who then compute their output locally. To avoid trivialities due to output size, we consider predicates, i.e., boolean functions $f : \{0,1\}^{Ln} \to \{0,1\}$.

We start by showing, for any predicate f, the existence of a deterministic protocol with certain properties given a randomized protocols with certain (other) properties.

Lemma 3.1. *Let f be a predicate. If there exists a randomized protocol that $R(t,p)$-solves f, then for any constant $c > \frac{1}{p}$ there exists a deterministic protocol that $D(c \cdot t, p - \frac{1}{c})$-solves f.*

Proof. Let P be a randomized protocol that $R(t,p)$-solves f. Using P, we construct another randomized protocol P' that works as follows. For the first ct rounds, P' operates exactly as P. At the end of round ct, each player that has not yet written on its output tape, writes an arbitrary value on the output tape. Note that on any input, by the Markov inequality, the probability that P runs more than ct rounds is at most $\frac{1}{c}$. It follows that, for any input $\vec{x}$, protocol P' has a deterministic upper bound of ct rounds on its running time, and that on any input $\vec{x}$, P' is correct with probability at least $p - \frac{1}{c}$. Next, we claim that there exists some specific random tape for P' for which the resulting deterministic protocol P^* $D(ct, p - \frac{1}{c})$-solves f. To see that such a tape exists, consider the uniform distribution over the inputs. Choose at random a random tape T for P'. The probability that P' with T is correct on a uniformly at random chosen input is at least $p - \frac{1}{c}$. By the pigeonhole argument this implies that there is a random tape T^* with which P' is correct with probability at least $p - \frac{1}{c}$ (over the uniformly at random chosen inputs). Since the inputs were taken from a uniform distribution, this implies that a $p - \frac{1}{c}$ fraction of inputs is computed correctly, and hence P^*, which is P' with tape T^*, $D(ct, p - \frac{1}{c})$-solves f. □

Lemma 3.2. *For every constant $p > 1/2$, the number of predicates $f : \{0,1\}^{Ln} \to \{0,1\}$ that admit a deterministic $D(L/B - 1, p)$-protocol is $o(2^{(2^{Ln})})$.*

Proof. A protocol is completely specified by a set of functions $\pi^\tau(i,j)$, $\tau \geq 1$, $0 \leq i,j \leq n-1$, which specify, for each round τ each sender i and each receiver j, what message is sent from i to j at round τ. We use $\pi^\tau(i,i)$ to denote the function that says, for each player $\mathcal{P}_i$ and round τ, what $\mathcal{P}_i$ writes on its output tape at round τ.

Now, any function $\pi^\tau(i,j)$ depends on the $(\tau-1)(n-1)$ messages received by player i until (and including) round $\tau - 1$, and on the input to player $\mathcal{P}_i$. It follows that any function $\pi^\tau(i,j)$ depends on

[1] Recall that typically, $L = \Theta(n \log n)$ and $B = \Theta(\log n)$.

at most $\tau(n-1)B+L$ bits. Also, $\pi^\tau(i,j) \in \{0,1\}^B$ for $i \neq j$ and $\pi^\tau(i,i) \in \{0,1,\perp\}$ (where $\perp$ means "no output"). Therefore there are at most $(2^B)^{2^{L+\tau(n-1)B}}$ possible functions $\pi^\tau(i,j)$.

Any round τ is defined by n^2 such function, therefore the number of different protocols that run in at most t rounds (i.e., the number of different $D(t,p)$ protocols to compute a boolean function) is bounded from above by

$$\prod_{\tau=0}^{t}\left((2^B)^{2^{L+\tau(n-1)B}}\right)^{n^2} = 2^{B\cdot n^2\sum_{\tau=1}^{t}2^{L+B\tau(n-1)}} \leq 2^{2Bn^2\cdot 2^{L+Bt(n-1)}}.$$

For $t \leq L/B-1$, and since $B<L$, this is $2^{o(2^{nL})}$.

Every given protocol P gives some output $\vec{y}=P(\vec{x})$ on any input $\vec{x}$. If the protocol P $D(t,p)$-solves f, then $|\{\vec{x}\in\{0,1\}^{Ln} : P(\vec{x}) \neq f(\vec{x})\}| \leq (1-p)2^{Ln}$. For a given protocol P the number of functions f that conform to this bound is at most

$$\sum_{d=0}^{\lfloor(1-p)2^{Ln}\rfloor}\binom{2^{Ln}}{d} \leq 2^{H(1-p)\cdot 2^{Ln}},$$

where H is the binary entropy function (this is an upper bound on the volume of a binary Hamming ball of radius $(1-p)2^{Ln}$).

It follows that the number of boolean functions f for which there exists a protocol that $D(L/B-1,p)$-solves f is at most

$$2^{(H(1-p)+o(1))\cdot 2^{Ln}},$$

and, since $H(1-p)<1$ for $p>1/2$, the lemma follows. □

We can now prove that most predicates do not admit a deterministic protocol with less than $\lceil L/B\rceil - 1$ rounds, even if the protocol is allowed to err on many (i.e., almost half of the) inputs. Formally, we prove the following theorem.

Theorem 3.3. *Suppose that each player has an L-bit input, and each link can carry a B-bit message in each round. Let $p>1/2$ be a constant. For a $(1-o(1))$-fraction of the predicates $f:\{0,1\}^{Ln}\to\{0,1\}$, there is no deterministic protocol that $D(L/B-1,p)$-solves f.*

Proof. The number of predicates $f:\{0,1\}^{Ln}\to\{0,1\}$ is

$$2^{(2^{Ln})}.$$

Hence, by Lemma 3.2, the fraction of predicates f for which there exists a protocol that $D(L/B-1,p)$-solves f, out of all predicates, is $o(1)$. □

Using Lemma 3.1 and Theorem 3.3 we have the following theorem.

Theorem 3.4. *Suppose that each player has an L-bit input, and each link can carry a B-bit message in each round. Let $p>1/2$ be a constant. For a $(1-o(1))$-fraction of the predicates $f:\{0,1\}^{Ln}\to\{0,1\}$, it holds that if P is a randomized protocol that $R(t,p)$-solves f, then $t=\Omega(L/B)$.*

We summarize with a corollary about the typical values of L and B.

Corollary 3.5. *Consider the predicates with $O(n \log n)$ bits of input at each player. Assume that in each round, each player sends to any other player a message of size $O(\log n)$ bits. Then for any constant $p > 1/2$, for most predicates f it holds that if a (randomized) protocol P $R(t,p)$-solves f, then $t = \Omega(n)$.*

4 Protocol for Pointer Chasing

In this section we consider the problem of *pointer chasing*, and study it in our model for the case of $B = \log n$ (i.e., each link can transmit a message of $\log n$ bits in each round).

We start with a formal definition of the problem.

4.1 The problem

The problem is defined for any integer $n > 0$. Each of the players $\mathcal{P}_i$, for $0 \leq i \leq n-1$, gets as input a vector A_i of size n, in which each entry holds a number in the range 1 to n. The interpretation of $A_i[j]$, for $1 \leq j \leq n$, is a pointer to the $A_i[j]$'th entry in the vector A_{i+1}. The goal is to compute for any $j \in [n]$, the value of $A_{n-1}[A_{n-2}[\ldots A_0[j]\ldots]]$. Intuitively, the goal is to follow a chain of pointers as in Fig. 1. We consider this problem on the clique when the bandwidth of each link is $\log n$ bits, i.e., when we set $B = \log n$ in our model.

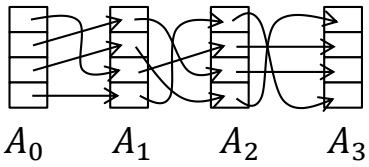

Figure 1: An illustration of the pointer chasing problem. Player i holds array A_i, and the goal is to follow the chain starting at a given position. Note that while this example demonstrates a case where the arrays A_i are permutations, this is not necessarily the case in all instances.

For ease of discussion and notations, the input can also be viewed as representing a layered directed graph of $n+1$ layers of width n, where the entry $A_i[j]$, for $0 \leq i \leq n-1$, $1 \leq j \leq n$ represents a directed edge from node j in layer i to node $A_i[j]$ in layer $i+1$. Note that every node has out degree exactly 1 (except layer n where the nodes have out degree 0). The task of the players, under this interpretation, is to compute, for every $1 \leq j \leq n$, the identity of the node of layer n which is reached by a length-n path starting at node j of layer 0. Note that the goal of the players is to compute a function (the output of all players is the same, i.e., a vector of length n, representing the nodes at the end of the paths).

4.2 Routing Primitives

To define our protocol we employ the following result of Lenzen (Theorem 3.7, [8]) for the *Information Distribution Task*.

Theorem 4.1 ([8]). *Consider a communication network in the form of a clique, where every link can transmit $O(\log n)$ bits per round. Any set of messages (of $O(\log n)$ bits each) such that each node sends at most n messages and each node has to receive at most n messages, can be routed distributively in $O(1)$ rounds.*

In our description of the protocol we use the procedure that is guaranteed by Theorem 4.1 and denote it ROUTE.

In addition to ROUTE we use another routing primitive that we call DISSEMINATE. For this routing primitive the players are partitioned into disjoint "teams" T_w. We assume that each player knows the complete partition of the players into teams. Further, in each teams T_w, there is a single *sender* $s_w \in T_w$ which has a vector v_i of n integers in the range $[1..n]$ to disseminate to all the members of its team. This primitive is implemented in two rounds as follows.

In the first round of DISSEMINATE each sender s_w sends to player $\mathcal{P}_t$, for $0 \leq t \leq n-1$, a message containing the tuple $(w, v_i[t+1])$, i.e., the identity of the team and the value of the $t+1$'th entry of the vector to disseminate. This can be done in parallel for all teams, in a single round using the links connecting the senders to the various players $\mathcal{P}_t$, for $0 \leq t \leq n-1$. In the second round of DISSEMINATE each player $\mathcal{P}_t$, $0 \leq t \leq n-1$ sends the messages it received to the appropriate players that need to get them, i.e., to the members of the appropriate team. That is, each player $\mathcal{P}_t$, $0 \leq t \leq n-1$ sends each message $(w, v_i[t+1])$ that it received to all players of the team T_w. This can be done in parallel for all players $\mathcal{P}_t$, $0 \leq t \leq n-1$, and all teams in a single round since each player sends a single message to any other player.

4.3 An $O(\log n / \log\log n)$-round protocol

For ease of presentation we describe the protocol in the terminology of solving the pointer chasing problem defined for layered graphs. Furthermore, for ease of presentation we assume that n is a power of 2.

The protocol starts with an input such that each player $\mathcal{P}_i$, for $0 \leq i \leq n-1$, holds a vector A_i of size n that represents the n endpoints of paths of length 1 edge each (i.e. single edges), starting in each of the n nodes of level i in the graph. The protocol works in iterations where in each iteration the endpoints of paths of longer and longer length are computed. This is done in an aligned way. That is, at the end of each iteration j there is a value ℓ_j such that the endpoints of all paths of length ℓ_j, emanating from all nodes in the graph of level i, such that $i \mod \ell_j = 0$, are known to player $\mathcal{P}_i$. All players $\mathcal{P}_i$ hold a vector $\vec{v_i}$ which is maintained to hold these endpoints. For the beginning of the protocol, i.e., at the end of the "empty iteration" $j = 0$, $\ell_j = 1$ and the vector $\vec{v_i}$ is initialized to the the input vector A_i of each player $\mathcal{P}_i$.

The protocol proceeds in two stages, which differ in the way that shorter paths are combined into

longer paths, and in the speed at which the path length grows.

The **first stage** is just the classical pointer doubling procedure run for r iterations (where r is to be determine later). Specifically, in iteration j, for every i such that $i \equiv 0 \pmod{2^j}$, $\mathcal{P}_{i+2^{j-1}}$ sends $A_{i+2^{j-1}}$ to player $\mathcal{P}_i$. Note that using the procedure ROUTE this can be done in each iteration in parallel for all players, in constant number of rounds.

By the end of the first stage, each player $\mathcal{P}_i$ with $i \equiv 0 \pmod{2^r}$ holds the endpoints of n paths of length 2^r edges each, emanating from level i in the graph. We then partition the players into teams of size 2^r players each, where each team T_w, $0 \leq w < n/2^r$ consists of the consecutive players $\mathcal{P}_{w2^r+j}$, for $0 \leq j < 2^r$. With these teams, and players $\mathcal{P}_{w2^r}$ as the sender for team T_w, we invoke procedure DISSEMINATE (see Section 4.2) to disseminate the endpoints known to the senders to all the members of their respective teams. This takes two rounds and ends the first stage of the protocol.

In the **second stage**, the algorithm proceeds by assigning tasks to *teams* of players. When an iteration of the second stage starts, all players in a team hold the same information. The goal of an iteration is that q short paths will be combined into a longer path (q to be determined later). This is done by way of trying out *all* possible concatenations of q consecutive paths. Each such concatenation is checked by some player, which verifies whether the ending node of each shorter path in the possible concatenation is equal to the starting node of the following path of the same possible concatenation. To do that, each one of the n^q possible concatenations of shorter paths is assigned to some player, which gets the appropriate information and verifies if the concatenation is valid. The parameter q is chosen so that the information needed by any player in order to check all concatenations assigned to it can be received in a constant number of rounds (using ROUTE), and that there are enough players to assign all n^q possible concatenations.

More precisely, let $C \stackrel{\text{def}}{=} 2^r$. The players are partitioned into teams, where each team consists of C consecutive players, i.e., each team T_w, $0 \leq w < n/2^r$ consists of the players $\mathcal{P}_{w2^r+j}$, for $0 \leq j < 2^r$. When iteration $k \geq 1$ of the second stage starts, we have $\frac{n}{C} \cdot q^{-(k-1)}$ *active teams*. Initially, when the second stage starts, all teams are active teams. The players of any active team all hold the endpoints of the n $(C \cdot q^{k-1})$-length paths emanating from the nodes of the layer that corresponds to the first player of the team (for the basis of the induction, the reader can verify that this holds for iteration $k = 1$ of the second stage, i.e., when the first stage of the protocol ends). We then take each q consecutive active teams as a *group*, and let each group find the endpoints of the paths of length $C \cdot q^k$ emanating from the layer that corresponds to the first player of the first team.

This is done as follows, in parallel in all groups. The players of the first team in the group are designated as *tester players*. Each such tester player receives, from each of the q teams, n/q endpoints of paths (of length $\frac{n}{C} \cdot q^{-(k-1)}$ each). It can then check all possible $(\frac{n}{q})^q$ possible combinations to possibly find valid concatenations. Since the tester team is composed of C players, we can check in this way $C \cdot (\frac{n}{q})^q$ possible concatenations. So long as $q^q \leq C$, this allows us to check all necessary n^q possible concatenations. Note that each tester node receives n endpoints and that, since each team is composed of C players, each player has to send n/q endpoints to some tester player. Therefore this procedure can be performed in $O(1)$ rounds using ROUTE.

When the information arrives as the tester nodes, they check the possible concatenations under their "responsibility." Once the nodes have terminated validating concatenations, each node that has

found valid concatenation(s) sends them to the first player of its team. Observe that exactly n valid concatenations will be found overall (per group) and that each player therefore identifies at most n valid concatenations. This can therefore be done in $O(1)$ rounds for all groups in parallel using ROUTE.

We then invoke the procedure DISSEMINATE so that all the endpoints known to the first player in each tester team is known to all members of that team. Observe that this is done in parallel for all tester teams in two rounds.

The tester teams of iteration k (of the second stage) then become the active teams for the next iteration, i.e., iteration $k+1$ of the second stage.

The **protocol ends** when there is a single active team, which is equivalent to the condition that the ending nodes of n paths of length n edges, emanating from level 0 in the graph have been computed, and are known to player $\mathcal{P}_0$. This information can then be broadcast to all players in two rounds, using DISSEMINATE once again (with a single team consisting of all players, and the sender being $\mathcal{P}_0$).

4.4 Analysis

The correctness of the algorithm was already discussed during the description of the algorithm. In short, it follows from standard doubling technique arguments for the first stage, and from the fact the the size of each team is large enough to check all possible concatenations in each iteration of the second stage.

To analyze the time complexity, observe that each iteration of either the first or second stage takes $O(1)$ rounds. The time complexity of the protocol is therefore linear in the number of iterations.

Fact 4.2. *The first stage consists of r iterations.*

Fact 4.3. *The second stage has $\log_q(n/C)$ iterations.*

Since the correctness of the second stage of the protocol requires that $2^r = C \geq q^q$, we fix $q = \sqrt{\log n}$ and $r = \frac{1}{2}\sqrt{\log n} \cdot \log\log n$. We can now conclude with the following theorem.

Theorem 4.4. *There is a deterministic protocol for the pointer chasing problem with running time of $O(\log n / \log\log n)$ rounds.*

Proof. Using the above two facts, and since $C = 2^r$, the running time of the protocol is

$$\begin{aligned} O(r + \log_q(n/2^r)) &= O(\frac{1}{2}\sqrt{\log n} \cdot \log\log n + \log_{\sqrt{\log n}}(n/2^r)) \\ &\leq O(\frac{1}{2}\sqrt{\log n} \cdot \log\log n + \log_{\sqrt{\log n}} n) \\ &\leq O(\log n / \log\log n) \ . \end{aligned}$$

□

Acknowledgments

We thank Pierre Fraigniaud and Zvi Lotker for useful discussions.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Co., Reading, MA, 1974.

[2] B. Awerbuch. Complexity of Network Synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[3] J. Brody. The maximum communication complexity of multi-party pointer jumping. In *IEEE Conference on Computational Complexity*, pages 379–386. IEEE Computer Society, 2009.

[4] C. Damm, S. Jukna, and J. Sgall. Some bounds on multiparty communication complexity of pointer jumping. *Computational Complexity*, 7(2):109–127, 1998.

[5] A. Drucker, F. Kuhn, and R. Oshman. On the power of the congested clique model. In *Proc. 33rd Ann. ACM Symp. on Principles of Distributed Compting*, 2014. To appear.

[6] G. Kol and R. Raz. Interactive channel capacity. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *STOC*, pages 715–724. ACM, 2013.

[7] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[8] C. Lenzen. Optimal deterministic routing and sorting on the congested clique. In *Proc. 32nd ACM Symp. On Principles of Distributed Computing (PODC)*, pages 42–50, 2013. Also available as arXiv:1207.1852.

[9] C. Lenzen and R. Wattenhofer. Brief announcement: Exponential speed-up of local algorithms using non-local communication. In *Proc. 29th ACM Symp. On Principles of Distributed Computing (PODC)*, pages 295–296, 2010.

[10] C. Lenzen and R. Wattenhofer. Tight bounds for parallel randomized load balancing: extended abstract. In *Proc. 43th Ann. ACM Symp. on Theory of Computing*, pages 11–20, 2011.

[11] Z. Lotker, B. Patt-Shamir, E. Pavlov, and D. Peleg. Minimum-weight spanning tree construction in $o(\log \log n)$ communication rounds. *SIAM J. Comput.*, 35(1):120–131, 2005.

[12] N. Nisan and A. Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, Feb. 1993.

[13] B. Patt-Shamir and M. Teplitsky. The round complexity of distributed sorting. In *Proc. 30th ACM Symp. On Principles of Distributed Computing (PODC)*, pages 249–256, 2011.

[14] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[15] S. J. Ponzio, J. Radhakrishnan, and S. Venkatesh. The communication complexity of pointer chasing. *Journal of Computer and System Sciences*, 62(2):323 – 355, 2001.

[16] Y. Shiloach and U. Vishkin. An $O(\log n)$ parallel connectivity algorithm. *J. Algorithms*, 3(1):57–67, 1982.

[17] L. G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33:103–111, August 1990.