The Communication Complexity of Distributed Task Allocation

Andrew Drucker
Computer Science and AI Laboratory, MIT
Cambridge, MA 02139
adrucker@mit.edu

Fabian Kuhn
Dept. of Computer Science
University of Freiburg
79110 Freiburg, Germany
kuhn@informatik.uni-freiburg.de

Rotem Oshman*
Computer Science and AI Laboratory, MIT
Cambridge, MA 02139
rotem@csail.mit.edu

ABSTRACT

We consider a distributed task allocation problem in which m players must divide a set of n tasks between them. Each player i receives as input a set X_i of tasks such that the union of all input sets covers the task set. The goal is for each player to output a subset $Y_i \subseteq X_i$, such that the outputs $(Y_1, ..., Y_m)$ form a partition of the set of tasks. The problem can be viewed as a distributed one-shot variant of the well-known k-server problem, and we also show that it is closely related to the problem of finding a rooted spanning tree in directed broadcast networks.

We study the communication complexity and round complexity of the task allocation problem. We begin with the classical two-player communication model, and show that the randomized communication complexity of task allocation is $\Omega(n)$, even when the set of tasks is known to the players in advance. For the multi-player setting with $m = O(n)$ we give two upper bounds in the shared-blackboard model of communication. We show that the problem can be solved in $O(\log n)$ rounds and $O(n \log n)$ total bits for arbitrary inputs; moreover, if for any set X of tasks, there are at least $\alpha|X|$ players that have at least one task from X in their inputs, then $O((1/\alpha + \log m) \log n)$ rounds suffice even if each player can only write $O(\log n)$ bits on the blackboard in each round. Finally, we extend our results to the case where the players communicate over an arbitrary directed communication graph instead of a shared blackboard. As an application of these results, we also consider the related problem of constructing a directed spanning tree in strongly-connected directed networks and we show lower and upper bounds for that problem.

*Rotem Oshman was supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.

PODC'12, July 16–18, 2012, Madeira, Portugal.

Categories and Subject Descriptors

F.2.2 [**Analysis of Alg. and Problem Complexity**]: Nonnumerical Alg. and Problems—*comp. on discrete structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms

Algorithms, Theory

Keywords

task allocation, directed spanning trees, multiparty communication complexity, unidirectional links

1. INTRODUCTION

In many distributed systems, a large amount of work is performed quickly and efficiently by partitioning the work across the participants in the network. The "work" to be performed can range from computational work, such as simulating a complex physical environment or solving a complex optimization problem, to physical work, such as having robots travel to various locations to carry out assorted tasks. In all cases, effectively parceling out parts of the global goal to be performed by individual participants is key to the overall efficiency of the system. This problem has been studied in many forms and guises, from fault-tolerant task allocation (see [21]) to centralized and distributed scheduling (e.g., [8, 6, 18] and many others), from theoretical (e.g., the k-server problem [7]) to practical approaches [1].

In the current paper we study the *communication complexity* of task allocation, that is, the total number of bits that the participants need to exchange to allocate the tasks between themselves. We consider an abstract version of the problem, $\textsc{TaskAllocation}_{m,n}$, where m players must jointly perform a set of n tasks (we often assume that the set of tasks is $\{1, \ldots, n\}$). Each player i receives as input a set $X_i \subseteq \{1, \ldots, n\}$ of tasks that it is capable of performing. For example, in the case of robots performing tasks at different geographical locations, the player's input might consist of the set of locations requiring servicing that the robot can "see" with its sensors. The goal is for the players to partition the tasks between them: each player i must output a subset of tasks $Y_i \subseteq X_i$, such that $\bigcup_i Y_i = \{1, \ldots, n\}$ and $Y_i \cap Y_j = \emptyset$ for all $i \neq j$. To make the problem feasible, we consider only inputs $X_1, \ldots, X_m$ such that $\bigcup_i X_i = \{1, \ldots, n\}$, that is, there *exists* some partition that covers all the tasks. The players are charged for

communicating among themselves, but not for writing their output sets $Y_1, \ldots, Y_m$. We consider various models of communication between the players, from shared-blackboard to arbitrary strongly-connected communication networks.

The task allocation problem can be viewed as a restricted one-shot instance of the well-known k-server problem [7], where a centralized online algorithm assigns tasks to k servers, minimizing the total cost of servicing all tasks. In the k-server problem each (server, task) pair is associated with a *cost* for having the server perform the task, and moreover, tasks arrive continually and must be assigned in an online manner. In TaskAllocation, all tasks are initially known, and all have a cost of either 1 (if the task can be performed by the player) or ∞ (if it cannot). Partitioning the tasks between the players corresponds to finding a minimum-weight assignment of tasks to servers. To the best of our knowledge, the k-server problem has not been studied from the perspective of communication complexity, although distributed variants have been studied (e.g., [2, 3]). In the current paper we are not interested in competitive analysis, as the variant we consider is single-shot; extensions to weighted inputs and the online setting remain interesting directions for future work.

Our work raises several open problems, which we discuss in Section 9. In general, two-player communication complexity lower bounds have proven very useful in proving lower bounds on various distributed problems (e.g., [19, 4, 9, 16]). However, distributed computation is more accurately captured by *multi-player* communication games in the number-in-hand model, where each player knows its own input (contrast with the number-on-forehead model, where each player knows all the *other* players' input). The number-in-hand model was neglected by the communication complexity community for a time, but recently several new techniques have led to exciting advances (see [10, 11, 12]). We believe that the complexity of distributed computing in the *CONGEST* model, where bandwidth is restricted, can be analyzed in terms of a multi-player communication game. Importing problems from the distributed computing world into the communication complexity model raises issues which are not often considered in existing communication complexity lower bounds: *search problems*, where players are allowed to choose one of many possible outputs (e.g., electing a leader or reaching consensus);[1] *partial knowledge*, where each player needs to output only part of the answer (as exemplified in the TaskAllocation problem); and *unicast communication cost*, where we wish to charge players for the number of other players they communicate with, not just the total communication complexity as in the shared blackboard model. We believe that these issues yield new and interesting questions in multi-player communication complexity.

Contributions. Despite the apparent simplicity of the problem, TaskAllocation is rich enough to admit a strong lower bound: in Section 4 we show that even for two players (using public random coins), $\textsc{TaskAllocation}_{2,n}$ has a randomized communication complexity of $\Omega(n)$. We apply this bound in Section 5 to show that computing a rooted spanning tree in directed broadcast networks with diameter 2, where each message is restricted to B bits, requires $\Omega(n/B)$ rounds — even when the size of the network is fixed in advance and nodes have unique identifiers in the range $1, \ldots, n$. In Section 6 we study the communication complexity of $\textsc{TaskAllocation}_{m,n}$ in the classical multi-player setting where players communicate over a shared blackboard. We give two randomized algorithms: the first requires players to write large messages on the blackboard, but has an overall communication complexity of $O(n \log(n+m))$ and terminates in $O(\log m)$ rounds with high probability on any input. The second algorithm we give works well when no small number of players has to take care of a large set of tasks, a property that is formally captured by our definition of *task-player expansion* in Section 3. For inputs with task-player expansion α, our second randomized algorithm terminates in $O\big((1/\alpha + \log m)\log n\big)$ rounds (or better, see Theorem 6.6 in Section 6) and uses messages of size $O(\log(n+m))$. This can be shown to be optimal to within a $\operatorname{polylog}(m,n)$ factor. In Section 7 we extend our results to an arbitrary strongly-connected communication network between the players, with the size of individual messages bounded by B. We show that in networks of diameter D, $\textsc{TaskAllocation}_{m,n}$ can be solved in $O(D + \sqrt{m/B}\log m + n\log(n+m)/B)$ rounds, using a total of $O((m+n)\operatorname{polylog}(m,n))$ bits of communication.

[1] Many well-known lower bounds in communication complexity concern decision problems, but there are some cases where search problems play an important role, e.g., [13].

2. RELATED WORK

Scheduling and task allocation. There is a vast body of literature concerning scheduling, allocation, and related problems. Offline and online scheduling problems are a mainstay of combinatorial optimization; see [5] for a survey on multiprocessor scheduling, and [14] for a survey on the k-server problem. Distributed k-server is studied in [3], which gives a generic technique for translating centralized k-server algorithms into decentralized ones while maintaining good competitive ratio. The number of messages sent by the decentralized algorithm is counted in its cost, but messages can be unboundedly large, and indeed their size in [3] grows with the space requirements of the original centralized algorithm. To our knowledge, the communication complexity (in number of bits) of k-server has not been studied.

Distributed spanning tree in directed networks. Our motivation for studying the TaskAllocation problem arose from a result in [16], where two of the authors proved that counting (i.e., determining the number of nodes in the network) requires $\Omega(n/B)$ rounds in directed strongly-connected broadcast networks of diameter 2, where message size is limited to B bits. The specific network used in the lower bound of [16] only admits rooted spanning trees of constant depth, and hence the counting lower bound translates to a lower bound on finding a rooted spanning tree: if a rooted spanning tree can be found quickly, then counting could be solved in $O(1)$ additional rounds by summing up the tree, contradicting the $\Omega(n/B)$ lower bound. However, this lower bound crucially relies on the assumption that the size of the network is initially unknown to the nodes, and this did not seem to be the "reason" for the hardness of finding a spanning tree. In the current paper we use a lower bound on the TaskAllocation problem to obtain an $\Omega(n/B)$ lower bound on finding a rooted spanning tree even when the size n is known in advance.

Communication complexity. Strong communication complexity lower bounds are known for many interesting two-player problems; for example, for SetDisjointness, where

the players must determine whether their input sets are disjoint, a tight bound of $\Omega(n)$ is shown in [20]. For a comprehensive treatment of the subject we refer to [17]. The TaskAllocation problem differs from classical problems such as SetDisjointness in that it is not a decision problem; on a given input there may be many permissible outputs. In addition, we do not require any of the players to know the *global* answer, only which tasks they have claimed for themselves. The problem is therefore less constrained than traditional problems. In particular, the two-player $\Omega(n)$ lower bound we give in Section 4 does not follow directly from the $\Omega(n)$ lower bound on SetDisjointness: in the TaskAllocation$_{2,n}$ problem we promise the players that the inputs X_1, X_2 cover all the tasks, that is, $X_1 \cup X_2 = \{1, \ldots, n\}$. With this promise, SetDisjointness becomes trivial, as X_1, X_2 are disjoint iff their sizes add up to exactly n. Moreover, when the sets are not disjoint, an element in the intersection can be found using only $O(\log^2 n)$ bits of communication (by binary search). A similar phenomenon occurs with other problems, such as InnerProduct and GapHammingDistance. Therefore the lower bound we give in Section 4 is proven from first principles using an information-theoretic argument, rather than appealing to existing communication complexity lower bounds.

3. PROBLEM STATEMENT

Task allocation. The *distributed task allocation* problem, denoted TaskAllocation$_{m,n}$, is defined over a set T of $|T| = n$ tasks and a set V of $|V| = m$ players. We often assume that $T = \{1, \ldots, n\}$. Each player $v \in V$ receives an input set $X_v \subseteq T$, with the promise that $\bigcup_{v \in V} X_v = T$. The goal is for each player to output a set $Y_v \subseteq X_v$, such that for all $u, v \in V$ we have $Y_u \cap Y_v = \emptyset$, and moreover, $\bigcup_{v \in V} Y_v = T$ (that is, $\{Y_v \mid v \in V\}$ is a partition of T).

The input assignment $\{X_v \mid v \in V\}$ induces a bipartite graph $H = (V \dot\cup T, F)$, where the edges F are given by $F := \{(v, x) \in V \times T \mid x \in X_v\}$. We call H the *task-player graph*. For convenience, for each task $i \in T$, we let $V_i := \{v \in V \mid i \in X_v\}$ denote the set of players that have task i in their input.

Communication model. In order to solve a given task allocation instance the players in V must communicate. In the current paper we assume synchronous communication, i.e., the players proceed in synchronous rounds. We consider two models of communication:

- In the classical *shared blackboard* model, the players communicate by writing messages on a shared blackboard which is visible to all other players.
- In addition, we are interested in the following *general network* model: players communicate over an arbitrary (possibly directed) graph $G = (V, E)$. In each round, every player (represented by a node of G) can send one message of B bits to all its out-neighbors in G (that is, communication is by *local broadcast*). We assume that initially the players do not know anything about the graph G except possibly its size, i.e., the number of players.

The shared blackboard model is a special case of the general network model, obtained by choosing $G := K_m$ (the complete graph on m nodes).

We are interested in the following performance measures:

- *Total communication complexity*: the total number of bits ever sent or written on the blackboard during the protocol.
- *Round complexity*: how many rounds of communication are required (where in each round each player can send/write one message).
- *Message size*: how many bits the players send or write on the blackboard in each round. This parameter is usually specified as an external constraint, and we denote it by B.

Task-player expansion. The hardness of an instance of TaskAllocation depends on the properties of the input assignment, represented by the task-player graph H. In particular, we will show that the complexity depends on how well tasks can be distributed among the players, as formally captured by the following definition.

Definition 1 (Task-Player Expansion). *The* task-player expansion *of a task-player graph $H = (V \dot\cup T, F)$ is defined as*

$$\alpha(H) := \min_{T' \subseteq T} \frac{\left|\bigcup_{x \in T'} V_x\right|}{|T'|}.$$

Informally, when the task-player expansion is large, each set of tasks can be assigned to many different players; the problem is in some sense less constrained, which makes it easier to solve. The smallest value $\alpha(H)$ can take is $1/n$, which occurs when one player receives all the tasks in his input and the others receive nothing. The largest value, obtained when H is the complete bipartite graph, is m/n.

General definitions and notation. We conclude this section with a few general definitions and notation that will be used throughout the paper. For an integer $k \geq 1$, we denote by $[k]$ the set $[k] := \{1, \ldots, k\}$. Some of our results concern directed graphs. The graphs are always assumed strongly-connected; for a strongly-connected directed graph G, the diameter $D(G)$ is the length of the longest directed shortest path between any two nodes in G.

4. TWO-PLAYER LOWER BOUND FOR TASK ALLOCATION

We begin by analyzing the complexity of task allocation in the classic two-party model, where two players, Alice and Bob, wish to allocate n tasks between them. In this section we let TaskAllocation$_n$ stand for TaskAllocation$_{2,n}$, and we use U, V to denote the inputs to Alice and Bob respectively and A, B to denote Alice and Bob's outputs. It is assumed that $T = [n]$ and both players know n.

The tasks over which Alice and Bob "contend" are the ones in the intersection of their inputs, $U \cap V$; these tasks must be output by one player but not by both. If Alice does *not* output some task that she received in her input, then she must know that this task is in the intersection of the inputs, and that Bob will output it. The connection between task allocation and finding the intersection of the players' inputs is formalized in the following easy lemma:

Lemma 4.1. *Let (A, B) be a valid output on instance (U, V) of* TaskAllocation$_n$*; that is, $A \cup B = [n]$, $A \cap B = \emptyset$, $A \subseteq U$, and $B \subseteq V$. Then (1) $U \setminus A \subseteq U \cap V$ and $V \setminus B \subseteq U \cap V$; and (2) $U \cap V \subseteq (U \setminus A) \cup (V \setminus B)$.*

PROOF. For (1), let $x \in U \setminus A$ (the other inclusion is similar). In particular, then, $x \notin A$, and since $A \cup B = [n]$, we must have $x \in B$. But $B \subseteq V$, and therefore $x \in U \cap V$.

For (2), let $x \in U \cap V$. Since $A \cup B = [n]$ we have $x \in A \cup B$; assume w.l.o.g. that $x \in A$. Because $A \cap B = \emptyset$ we have $x \notin B$, but on the other hand we have $x \in V$ (as $x \in U \cap V$). Together we have $x \in V \setminus B$. □

THEOREM 4.2. *The randomized (public-coin) communication complexity of two-player task allocation is $\Omega(n)$.*

PROOF. Suppose the input (U, V) is generated according to the following distribution $\mathcal{D}$. Choose a random subset $X \subseteq [n]$ of size pn. Next, choose a random subset $Y \subseteq [n] \setminus X$ of size $(1-p)n/2$. Let $U := X \cup Y$ and let $V := X \cup \bar{Y} = \bar{Y}$. Let $\mathcal{C}$ be the support of the distribution (i.e., each set is of size $(1+p)n/2$, and the intersection is of size pn).

We are interested in the probability over $(U, V) \sim \mathcal{D}$ that given only U, Alice can guess $U \cap V = X$. Given U, the set X is a uniformly-chosen subset of pn elements of U. Therefore, given U, Alice's chance of guessing X is at best

$$\binom{(1+p)n/2}{pn}^{-1} \leq \left(\frac{(1+p)n/2}{pn}\right)^{-pn} = \left(\frac{1+p}{2p}\right)^{-pn}.$$

Informally, we will show that if there exists a protocol P for task allocation with communication complexity $o(n)$, then Alice can guess $X = U \cap V$ with statistically impossible accuracy (i.e., she can succeed with probability better than the bound above). For an execution of P, let A and B be the outputs of Alice and Bob, respectively. By Lemma 4.1, $U \setminus A \subseteq U \cap V$; in other words, after executing P, Alice knows that each element in her input that she did not output is in $U \cap V$. If $U \setminus A$ is large, this provides Alice with enough information to guess the remaining elements of $U \cap V$ "too accurately".

More formally, let P be a public-coin protocol for task allocation with t rounds, which succeeds with probability at least $1/2$ on each input. For each input $(U, V) \in \mathcal{C}$ we have $|U \cap V| = pn$ (by definition of $\mathcal{C}$), and by Lemma 4.1, $|(U \setminus A) \cup (V \setminus B)| \geq pn$ (in fact the lemma shows equality, but we do not require it here). Thus, if P succeeds then either $|U \setminus A| \geq pn/2$ or $|V \setminus B| \geq pn/2$; assume w.l.o.g. that with probability at least $1/4$ over both the choice of (U, V) and the coin tosses of P, the players eventually output a correct output (A, B) with $|U \setminus A| \geq pn/2$.

Now Alice can guess $X = U \cap V$ given U as follows: she simulates protocol P by guessing a t-bit transcript (in addition to P's own randomness), obtaining some output A. With probability at least $1/4 \cdot 2^{-t}$, Alice guesses a transcript for P that matches the input and P's randomness, and in addition, with this transcript and public randomness, P succeeds (so $U \setminus A \subseteq X$), and we have $|U \setminus A| \geq pn/2$. Now there are only at most

$$\binom{((1+p)n/2 - |U \setminus A|}{pn - |U \setminus A|} \leq \binom{(1+p)n/2}{pn/2} \leq \left(\frac{(1+p)en/2}{pn/2}\right)^{pn/2} = \left(\frac{(1+p)e}{p}\right)^{pn/2}$$

possibilities for X. By choosing the most likely possibility given the transcript and the public randomness, Alice can guess the correct value of X with probability at least

$$\left(\frac{(1+p)e}{p}\right)^{-pn/2} = \left(\frac{1+p}{2p} \cdot 2e\right)^{-pn/2}.$$

Therefore we must have

$$\frac{1}{4} \cdot 2^{-t} \cdot \left(\frac{1+p}{2p} \cdot 2e\right)^{-pn/2} \leq \left(\frac{1+p}{2p}\right)^{-pn}.$$

Simplifying yields

$$t \geq \frac{pn}{2}\left(\log \frac{1+p}{2p} - (1 + \log e)\right) - 2.$$

To obtain a non-trivial lower bound we must select p such that $\log \frac{1+p}{2p} > (1 + \log e) \approx 2.4$. For example, $p := 1/16$ satisfies this constraint.

To conclude, if P is a protocol for 2-player task allocation, then there must exist at least one input on which with probability at least $1/2$, at least $\frac{n}{32}\left(\log \frac{17}{2} - (1 + \log e)\right) - 2 = \Omega(n)$ bits are exchanged. Therefore the worst-case expected communication complexity of P is $\Omega(n)$. □

Remark 1. The lower bound can be extended to a relaxed variant of TASKALLOCATION, where we allow an ε-fraction of tasks to be assigned to *both* players for a sufficiently small constant $\varepsilon \geq 0$.

Remark 2. The two-player communication complexity of TASKALLOCATION$_n$ is $O(n)$, since Alice can always just send her complete input (represented as the n-bit characteristic vector) to Bob, claim all the tasks in her input, and have Bob claim the remaining tasks. Theorem 4.2 shows that this strategy is optimal. Moreover, if we wish to find all the elements in the intersection, when the intersection is of size $\Omega(n)$, repeatedly sampling a random element is the optimal strategy up to a $\log(n)$ factor.

5. LOWER BOUND ON FINDING ROOTED SPANNING TREES

Next we show how to apply the lower bound from the previous section to obtain a lower bound on computing a rooted spanning tree. Formally, the *distributed rooted spanning tree* problem in a network $G = (V, E)$ requires each node v in the network to output a value $p_v \in V \cup \{\perp\}$, such that the edges $\{(v, p_v) \mid v \in V\}$ form a rooted spanning tree (oriented upwards toward the root) of G. (Exactly one node v may output $p_v = \perp$, and this node is the root of the tree.) In each round of the algorithm, each node $v \in V$ broadcasts B bits, which are delivered to all of v's out-neighbors in G. Each node of G initially knows the size n of the graph and has a unique identifier (UID) drawn from the set $[n]$. [2]

Our lower bound shows that finding a rooted spanning tree is hard even in a restricted class $\mathcal{G}_n$ of networks, where each $G \in \mathcal{G}_n$ is strongly-connected, has a diameter of 2, and has no simple directed path of length more than 4. (In particular, all spanning trees have depth at most 4, so the algorithm cannot be "confused" by long paths or by tall potential spanning trees.)

[2]In [16], two of the authors proved a weaker version of this lower bound, which relied entirely on the assumption that the size of the network is not known a-priori. We now show that this assumption did not capture "the core hardness" of finding a rooted spanning tree; the problem remains hard without it.

THEOREM 5.1. *Any algorithm for finding a rooted spanning tree in networks of $\mathcal{G}_n$ requires at least $\Omega(n/B)$ rounds to succeed with probability $1/2$.*

PROOF. We prove the theorem by reduction to the two-party task allocation problem $\textsc{TaskAllocation}_{n-2}$. Specifically, we show that if there is an algorithm for finding a rooted spanning tree in all networks of $\mathcal{G}_n$ which requires t rounds to succeed with probability $1/2$, then there is a public-coin protocol for solving $\textsc{TaskAllocation}_{n-2}$ with communication complexity $O(B \cdot t)$. The theorem then follows from Theorem 4.2.

Fix an algorithm $\mathcal{A}$ for finding a rooted spanning tree. Given inputs U, V (respectively), Alice and Bob can solve $\textsc{TaskAllocation}_{n-2}$ by simulating the execution of $\mathcal{A}$ in a network $G_{U,V} = ([n], E_{U,V})$, where

$$E_{U,V} = (\{n-1, n\} \times [n-2]) \cup \{(n-1, n), (n, n-1)\} \cup (U \times \{n-1\}) \cup (V \times \{n\}) .$$

Informally, in $G_{U,V}$ nodes $n-1$ and n represent Alice and Bob respectively, and nodes $1, \ldots, n-2$ represent the task set of the $\textsc{TaskAllocation}_{n-2}$ problem. Nodes $n-1$ and n always have edges to all nodes of the network, regardless of the input. In addition, the nodes of U have edges to node $n-1$ (that is, to "Alice") and the nodes of V have edges to node n ("Bob"). It is easy to verify that $G_{U,V} \in \mathcal{G}_n$.

Alice and Bob cooperate to simulate the execution of $\mathcal{A}$, using the public randomness to assign outcomes to the coin tosses of nodes $1, \ldots, n$; bit $n \cdot k + i - 1$ of the public random string is interpreted as bit k of node i's randomness (for $i \in [n]$ and $k = 0, 1, 2, \ldots$). Alice is responsible for locally simulating nodes $U \cup \{n-1\}$; she keeps track of these nodes' states throughout the execution. Similarly, Bob is responsible for simulating nodes $V \cup \{n\}$. (The nodes in $U \cap V$ are simulated by both players independently.)

To simulate one round of $\mathcal{A}$, the players update the states of their locally-simulated nodes as follows: Alice computes the messages output by nodes $U \cup \{n-1\}$, and Bob computes the messages output by nodes $V \cup \{n-1\}$ in the current round; then Alice and Bob send each other the messages output by nodes $n-1$ and n (resp.). Now Alice computes the new state of each node in U after receiving the messages sent by nodes $n-1$ and n, and Bob does the same for the nodes in V. Finally, Alice computes the new state of node $n-1$ after receiving the messages sent by nodes $U \cup \{n\}$, and Bob updates the state of node n after receiving the messages of nodes $V \cup \{n-1\}$. Note that in the final step, Alice and Bob know *which* nodes' messages to deliver, because Alice knows U and Bob knows V. It is not hard to see that for nodes $i \in U \cap V$, Alice and Bob agree on the local state of i at every step of the simulation.

Suppose that $\mathcal{A}$ succeeds with probability at least $1/2$ after t rounds. Then after simulating round t of the execution, with probability at least $1/2$ each node $i \in [n-2]$ outputs a parent $p_i \in [n] \cup \{\perp\}$, with exactly one node $r \in [n]$ outputting $\perp$. The edges $\{(i, p_i)\}$ form a directed spanning tree with root r. To handle the root r, the protocol concludes with one final exchange: Alice sends Bob one bit b indicating whether some node $i \in U$ (that Alice was simulating) output $p_i = \perp$. Finally the players output the following sets:

$$A = \{i \in U \mid p_i = n-1\} \cup \{r, \text{ if } r \in U \text{ and } p_r = \perp\};$$
$$B = \{i \in V \mid p_i = n\} \cup \{r, \text{ if } r \in V \text{ and } p_r = \perp \text{ and } b = 0\}.$$

It is easy to verify that A, B form a valid output on instance (U, V), as each node in $[n-2]$ except possibly r must choose either $n-1$ or n as its parent (but not both), and if $r \in [n-2]$ then it is assigned to exactly one player.

The total amount of communication used by the protocol is $2Bt + 1 = O(Bt)$, and a correct output is produced with probability at least $1/2$. □

The lower bound above can be shown to be nearly-tight for networks of constant diameter (and in particular, the class $\mathcal{G}_n$). More generally, in networks of diameter D it is possible to construct a rooted spanning tree in $O(D^2 + n \log n/B)$ rounds, as we will see in Section 8. It is also easy to show that $O(D + |E|/B)$ rounds suffice for networks of any diameter. The time complexity of finding a spanning tree in networks with diameter $D = \omega(\sqrt{n})$ and $|E| = \omega(n)$ remains open to the best of our knowledge.

6. MULTIPARTY COMPLEXITY IN THE SHARED BLACKBOARD MODEL

In this section we study the complexity of distributed task allocation in the shared blackboard communication model. First note that the two-player lower bound (Thm. 4.2) can be embedded into the multi-player setting, yielding the following lower bound:

THEOREM 6.1. *The shared-blackboard communication complexity of multi-player task allocation is $\Omega(n)$. Further, for any $\alpha > 0$, $n \geq 2$, and $m \geq 2/\alpha$, there is a class of inputs with task-player expansion α for which some player needs to communicate $\Omega(1/\alpha)$ bits.*

PROOF. The 2-player scenario is a special case of the multi-player shared blackboard model. Therefore, it follows from Theorem 4.2 that the communication complexity of the multi-player task allocation problem is $\Omega(n)$.

For the bound involving the expansion α, choose two players $u, v \in V$, and partition the tasks into two sets $T', T \setminus T$, where $|T'| = 2/\alpha$. All the tasks in $T' \setminus T$ are assigned to each of the players in $V \setminus \{u, v\}$. As for the tasks in T', we use them to construct a random 2-player input for the player u and v as in Theorem 4.2. Because the sub-problem defined by $\{u, v\}$ and T' is statistically independent of the problem defined by the remaining players and tasks, u and v have to solve their sub-problem by themselves. Hence, by Theorem 4.2 either u or v has to communicate at least $\Omega(1/\alpha)$ bits. □

In the remainder of this section we give two algorithms for task assignment in the shared blackboard model. Our algorithms show that up to logarithmic factors, the bounds of the theorem above are tight.

In both of our algorithms, each round causes some player-task assignments to become fixed for the remainder of the algorithm (i.e., some tasks become *permanently assigned*). We let $T(i)$ denote the set of tasks that have not been permanently assigned by the beginning of round i, and $V(i)$ denote the set of players that still have some unassigned tasks at the beginning of round i. Further, let $H(i)$ be the subgraph of H (the task-player graph) induced by $V(i) \cup T(i)$. In a similar manner, we use $n(i) := |T(i)|$, $m(i) := |V(i)|$ and $X_v(i)$ to denote the number of remaining tasks, the number of remaining players, and player v's remaining (unassigned) tasks at the beginning of round i.

6.1 Large Messages, Small Total Complexity

We first give a randomized algorithm that tries to minimize the overall number of bits while keeping the number of rounds small at the same time. We do not restrict B, that is, individual players can potentially send large messages, as long as the total number of bits sent in by all players in all messages is not too large.

The algorithm proceeds as follows. In round $i \geq 1$, each player $v \in V(i)$ selects a subset of its remaining tasks $X_v(i)$: each task $x \in X_v(i)$ is selected independently with probability $\min\{1, 2^i/m\}$. Then, v proposes the assignments (v, x) for all selected tasks $x \in X_v(i)$, by writing these proposals on the shared blackboard. Each task that some player proposed to claim is assigned to the smallest player that attempted to claim it (note that this requires no further communication, as all players can see all proposals). This process continues until all tasks have been assigned.

In round $\lceil \log m \rceil$, each remaining player selects each of its remaining tasks with probability 1. Therefore the algorithm terminates in at most $\lceil \log m \rceil$ rounds. In the following, we show that with high probability, the total number of announced proposals is $O(n)$.

For a task $x \in T$, let A_x be the number of proposals of the form (v, x) that are announced throughout an execution of the algorithm. The following lemma analyzes the distribution of A_x for a task x.

LEMMA 6.2. *The random variable A_x is dominated by a constant multiple of a geometric random variable, that is, for $k \geq 1$, $\mathbb{P}(A_x \geq k) \leq c\rho^k$ for constants $c > 0$ and $\rho \in (0, 1)$.*

PROOF. Let $d = |V_x|$ be the degree of task x in H, i.e., the number of players that received task x in their input. The number of proposals made for each task depends on the round in which it is assigned: if task x is not assigned before round i, the number of proposals (v, x) in round i is binomially distributed with parameters d and $2^i/m$.

Let I be the round in which task x is assigned. Note that, since each task is permanently assigned in the first round where someone attempts to claim it, proposals (v, x) for task x are only made in round I. For $k \in \{0, \ldots, d\}$ and $i \in [\lceil \log m \rceil]$, we define

$$p(k, i) := \mathbb{P}(A_x = k | I = i) = \binom{d}{k} \frac{2^{ik}}{m^k} \left(1 - \frac{2^i}{m}\right)^{d-k}.$$

We can express the probability distribution of A_x in terms of these probabilities as follows. For $k \in \{0, \ldots, d\}$, we have

$$\begin{aligned}
\mathbb{P}(A_x = k) &= \sum_{i=1}^{\lceil \log m \rceil} p(k, i) \cdot \mathbb{P}(I = i) \\
&= \sum_{i=1}^{\lceil \log m \rceil} p(k, i) \cdot \prod_{j=1}^{i-1} p(0, j) \\
&= \sum_{i=1}^{\lceil \log m \rceil} p(k, i) \cdot \prod_{j=1}^{i-1} \left(1 - \frac{2^j}{m}\right)^d \\
&\leq \sum_{i=1}^{\lceil \log m \rceil} \underbrace{\frac{1}{k!} \cdot \left(\frac{d2^i}{m}\right)^k \cdot \prod_{j=1}^{i-1} \left(1 - \frac{2^j}{m}\right)^d}_{q(k,i)}. \quad (1)
\end{aligned}$$

In the last inequality we used the fact that $\binom{d}{k} \leq \frac{d^k}{k!}$. Let us consider the values of the expression $q(k, i)$ from Inequality (1) for a fixed k and different i. Let i_0 be the maximal value for i such that $2^i \leq m/d$. We have $q(k, i_0) \leq 1/k!$ and $q(k, i) \leq 2^{-(i_0 - i)k} q(k, i_0)$ for $i < i_0$. Because $k \geq 1$, we therefore get

$$\sum_{i=1}^{i_0} q(k, i) \leq \sum_{i=1}^{i_0} \frac{1}{k!} \cdot 2^{-(i_0 - i)} \leq \frac{2}{k!}. \quad (2)$$

Further, we obtain $q(k, i_0 + 1) \leq 2^k/k!$, and for $i > i_0 + 1$ we have

$$\begin{aligned}
q(k, i) &\leq q(k, i_0 + 1) \cdot 2^k \cdot \left(1 - \frac{2^{i-1}}{m}\right)^d \\
&\leq q(k, i_0 + 1) \cdot 2^k \cdot e^{-\frac{2^{i-1} d}{m}} \leq q(k, i_0 + 1) \cdot \left(\frac{2}{e^{2^{i - i_0 - 1}}}\right)^k.
\end{aligned}$$

In the first inequality, we used the fact that $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$, and in the second inequality the fact that $i > i_0 + 1$ and $k \leq d$. Combining with (2) and applying (1), we get that

$$\mathbb{P}(A_x = k) \leq \sum_{i=1}^{\lceil \log m \rceil} q(k, i) = O\left(\frac{2^k}{k!}\right).$$

The claim of the lemma now follows because $\mathbb{P}(A_x \geq k) = \sum_{k'=k}^{d} \mathbb{P}(A_x = k')$. □

The total number of assignment proposals throughout the execution is $A = \sum_{x \in T} A_x$. The random variables A_x are independent, and by Lemma 6.2, the sum A can be bounded from above (up to a constant factor) by the sum of n independent geometric variables. Thus we obtain the following:

LEMMA 6.3. *In an execution of the above algorithm, the total number of announced potential assignment (v, x) for $v \in V$ and $x \in T$ is at most $O(n) = O(|T|)$ with probability at least $1 - e^{-cn}$ for any constant $c > 0$.*

PROOF. Let A be the total number of announced potential assignments (v, x). By the definition of the random variables A_x, we have $A = \sum_{x \in T} A_x$. Note further that the random variables A_x are independent because they depend on disjoint set of edges of the player-task graph H and edges are picked independently.

By Lemma 6.2, each A_x is dominated by a $c \cdot Y_x$ for a constant $c > 0$ and a geometric random variable $Y_x \sim \text{Geom}(p)$ for a constant parameter $p \in (0, 1)$. Consequently, A is dominated by $c \cdot Y$, where Y is the sum of n independent geometric random variables with parameter p. Let $k > 0$ be a positive integer and let $Z \sim \text{Bin}(k, p)$ be a binomial random variable with parameter k and p. We have $Y > k$ iff in a sequence of k Bernoulli trials with success probability p, less than n succeed. We therefore have $\mathbb{P}(Y > k) = \mathbb{P}(Z < n)$. If we choose $k = \gamma n/p$ for a constant $\gamma > 1$, we have $\mathbb{E}[Z] = \gamma n$ and therefore $Z \geq n$ with probability at least $1 - e^{-\Omega(n)}$, where the hidden constant in the exponent can be made arbitrarily large if the constant γ is chosen sufficiently large. □

The performance of the algorithm follows directly from the lemma above:

THEOREM 6.4. *The above algorithm solves the task assignment problem in the shared blackboard model in $O(\log m)$ rounds and with an overall communication complexity of $O(n \log(n + m))$ bits.*

6.2 Task Allocation with Small Messages

The first algorithm we presented is efficient in terms of total bit complexity and number of rounds. However, it might require individual players to send a large number of bits in a single round. We now consider the case where in each round, each player can only send a message of at most $B = O(\log(n+m))$ bits. We give an algorithm that has a good round complexity when the expansion $\alpha(H)$ of the task-player graph is large (cf. Definition 1). In the following, we assume that $\alpha(H) \leq 1$, i.e., the number of players does not exceed the number of tasks. (If $\alpha(H) > 1$, all appearances of $\alpha(H)$ in our bounds can be replaced by 1.)

Description of the algorithm. As before, the algorithm runs in rounds, and we let $H(i) = (V(i) \cup T(i), F(i))$ be the remaining task-player graph at the beginning of round i. In each round, every player v picks a random task $x \in X_v(i)$ uniformly, and proposes the assignment (v, x) by writing it on the blackboard. Task x is then permanently assigned to the smallest player u that attempted to claim it (i.e., that wrote (u, x) on the blackboard). Unassigned tasks $y \in T(i)$ for which that no assignment (v, y) was proposed in round i remain unassigned. We continue until all tasks in T have been assigned to some player.

Analysis of the running time. The algorithm above makes progress in one of two ways. Let $\lambda \in (0, \alpha(H))$ be a parameter whose value will be fixed later.

Definition 2 (Task-reducing rounds). *We say that round i is a* task-reducing round *if given the random choices up to the beginning of round i, the expected number of tasks assigned in round i is at least $\lambda|V(i)|$.*

Definition 3 (Edge-reducing rounds). *Round i is called* edge-reducing for player $v \in V(i)$ *if given the random choices up to the beginning of round i, in expectation at least $\left(1 - \sqrt{\lambda/\alpha(H)}\right) \cdot |X_v(i)|$ tasks from $X_v(i)$ are permanently assigned in round i.*

Informally, if a round is task-reducing, we make progress because many tasks become assigned. On the other hand, if the round is not task-reducing, this means that many players picked the same task to propose (because each player proposes one task, but not many tasks were proposed in total). Each task x proposed in round i becomes assigned to some player, and the other players v then remove this task from their remaining input $X_v(i)$, causing edge (v, i) to be removed from $H(i)$. If H has good expansion, many players are incident to (i.e., have in their input) some task among the tasks proposed in round i, and all such players now shed all edges corresponding to proposed tasks. Therefore the round is edge-reducing for a good fraction of players.

More formally, we prove the following lemma.

Lemma 6.5. *For each round i of the algorithm, either round i is a task-reducing round, or round i is an edge-reducing task for at least a $\left(1 - \sqrt{\lambda/\alpha(H)}\right)$-fraction of the remaining players $v \in V(i)$.*

Proof. Let $S \subseteq T(i)$ be a random variable representing the number of tasks that are assigned in round i, given the random choices up to the beginning of the round. If round i is not task-reducing, then $\mathbb{E}[S] < \lambda|T(i)|$. We will show that in this case round i is edge-reducing for a large fraction of remaining players. In the sequel all probabilities and expectations are implicitly conditioned on events up to the beginning of round i.

Let C_u be the event that player $u \in V(i)$ picks a task $x \in X_v(i)$ that is also picked by another player $v \in V(i)$. From the assumption that the round is not task-reducing,

$$\sum_{u \in V(i)} \mathbb{P}(C_u) > |V(i)| - \mathbb{E}[S] > |V(i)| - \lambda|T(i)|$$
$$\geq |V(i)| \cdot \left(1 - \frac{\lambda}{\alpha(H)}\right). \tag{3}$$

The last inequality follows because $|V(i)| \geq \alpha(H)|T(i)|$, by the assumption that H has task-player expansion $\alpha(H)$.

For $u \in V(i)$, let Z_u be the number of tasks in $X_u(i)$ that are assigned in round i, and let Z'_u be the number of tasks in $X_u(i)$ that are proposed by other players $v \in V(i) \setminus \{u\}$. Clearly, $Z'_u \leq Z_u$, and therefore also $\mathbb{E}[Z'_u] \leq \mathbb{E}[Z_u]$. We have

$$\mathbb{P}(C_u) = \sum_x \mathbb{P}(Z'_u = x) \cdot \mathbb{P}(C_u | Z'_u = x)$$
$$= \sum_x \mathbb{P}(Z'_u = x) \cdot x = \mathbb{E}[Z'_u] < \mathbb{E}[Z_u].$$

We need to show that for at least $\left(1 - \sqrt{\lambda/\alpha(H)}\right)|V(i)|$ players $u \in V(i)$ we have $\mathbb{E}[Z_u] \geq \left(1 - \sqrt{\lambda/\alpha(H)}\right) \cdot |V(i)|$ (i.e., round i is edge-reducing for these players). Suppose not. Then

$$\sum_{u \in V(i)} \mathbb{P}(C_u) < \sum_{u \in V(i)} \mathbb{E}[Z_u]$$
$$< \left(1 - \sqrt{\frac{\lambda}{\alpha(H)}}\right) |V(i)| \cdot 1$$
$$+ \sqrt{\frac{\lambda}{\alpha(H)}} \cdot |V(i)| \cdot \left(1 - \sqrt{\frac{\lambda}{\alpha(H)}}\right)$$
$$= \left(1 - \frac{\lambda}{\alpha(H)}\right) \cdot |V(i)|,$$

a contradiction to Inequality (3). □

To see the intuition behind the algorithm's progress, consider the simple case where λ and $\alpha(H)$ are both constant. Then each task-reducing round causes a constant fraction of tasks to be eliminated, and each edge-reducing round causes a constant fraction of players to shed a constant fraction of their edges in the task-player graphs. After roughly $\log(n)$ edge-reducing rounds, a constant fraction of players have no tasks remaining, and they are removed from $V(i)$. To eliminate all players (and hence all tasks) we require logarithmically-many such "phases", so the overall time complexity is $O(\log n \cdot \log m)$.

In the following theorem we obtain a slightly better bound by carefully setting the parameter λ:

Theorem 6.6. *With high probability, the algorithm runs in at most T rounds, where*

$$T = O\left(\frac{\log m \log n}{\log^2(\alpha(H) \log m)}\right) \quad \text{if } \alpha(H) = \Omega\left(\frac{1}{\log m}\right), \text{ and}$$
$$T = O\left(\frac{\log n}{\alpha(H)}\right) \quad \text{if } \alpha(H) = O\left(\frac{1}{\log m}\right).$$

PROOF. Let $\lambda \leq \alpha(H)/5$ be a positive parameter. The value of λ will be fixed later. Consider some round i. We call i a task-reducing round if the expected number of tasks assigned in round i is at least $\lambda|V(i)|$. For a player $v \in V(i)$, we call round i an edge-reducing round for v if in expectation at least $\big(1-\sqrt{\lambda/\alpha(H)}\big)\cdot|X_v(i)|$ tasks from $X_v(i)$ are assigned to some player in round i. By Lemma 6.5, each round i is either a task-reducing round or an edge-reducing round for at least a $\big(1-\sqrt{\lambda/\alpha(H)}\big)$-fraction of the players in $V(i)$.

Let us first look at a task-reducing round i. For a task $x \in T(i)$, let S_x be an indicator random variable that is 1 iff task x is assigned in round i. The sum $S = \sum_{x\in T(i)} S_x$ then counts the number of tasks that are assigned to some player in round i. The picking of tasks by players can be seen as a balls-into-bins process in which each ball (player) independently chooses a random bin according to some distribution.

Consider two tasks $x \neq y \in T(i)$ and a player $v \in V(i)$ and let A_{vx} be the event that player v picks task x. We have $\mathbb{P}(A_{vx}) = 1/|X_v(i)|$. If we condition on $S_y = 0$, we have $\mathbb{P}(A_{vx}|S_y = 0) = 1/(|X_v(i)|-1) > \mathbb{P}(A_{vx})$ if $y \in X_v(i)$ and $\mathbb{P}(A_{vx}|S_y = 0) = \mathbb{P}(A_{vx})$ otherwise. We therefore have $\mathbb{P}(S_x = 1|S_y = 0) \geq \mathbb{P}(S_x = 0)$ and thus $\mathbb{P}(S_x = 1|S_y = 1) \leq \mathbb{P}(S_x = 1)$. Consequently it holds that $\mathbb{E}[S_xS_y] \leq \mathbb{E}[S_x]\cdot\mathbb{E}[S_y]$, i.e., S_x and S_y are negatively correlated. We therefore get $\mathrm{Var}(S) \leq \sum_{x\in T(i)} \mathrm{Var}(S_x) = \sum_{x\in T(i)} \mathbb{P}(S_x = 1)\big(1-\mathbb{P}(S_x = 1)\big) < \sum_{x\in T(i)} \mathbb{E}[S_x] = \mathbb{E}[S]$. Note that It therefore follows from Chebyshev's inequality (and because $S \geq 1$ in any case) that $S \geq \lambda|T(i)|/2$ with at least constant probability.

Let us now consider a round i that is edge-reducing for player $v \in V(i)$. Let Z_v be the number of edges of player v after round i. By assuming the round i is edge-reducing for v, we have $\mathbb{E}[Z_v] \leq \sqrt{\lambda/\alpha(H)}\cdot|X_v(i)|$. Hence, by applying the Markov inequality to Z_v, we get that

$$\mathbb{P}\left(Z_v > \left(\frac{\lambda}{\alpha(H)}\right)^{1/4}\cdot|X_v(i)|\right) < \left(\frac{\lambda}{\alpha(H)}\right)^{1/4}. \qquad (4)$$

Assume that there are $k = c\log_{\alpha(H)/\lambda}(n)$ edge-reducing rounds for player v for a sufficiently large constant c. Let Y be the number of these k rounds in which more than a $\sqrt[4]{\lambda/\alpha(H)}$-fraction of v's tasks remain. By (4), Y is dominated by a binomial random variable with parameters k and $\sqrt[4]{\lambda/\alpha(H)}$. As long as $Y < k - 4\log_{\alpha(H)/\lambda}(n)$, all tasks of player v get assigned to some player during these k rounds. Using a standard Chernoff bound, we get that for c sufficiently large, $Y < k - 4\log_{\alpha(H)/\lambda}(n)$ with high probability. Therefore, as soon as there are $c\log_{\alpha(H)/\lambda}(n)$ edge-reducing rounds for each player $v \in V$, all tasks have been assigned. We next show that this has to be the case after $O(\log_{\alpha(H)/\lambda}(n)\cdot\log_{\alpha(H)/\lambda}(m))$ rounds i such that round i is edge-reducing round for at least a $\big(1-\sqrt{\lambda/\alpha(H)}\big)$-fraction of the players in $V(i)$.

As before, let $k = c\log_{\alpha(H)/\lambda}(n)$ be the number of edge-reducing rounds for a player v needed to assign all tasks of v w.h.p. Consider the state at the beginning of some round i_0 and assume that there are $2k$ rounds that are edge-reducing for at least a $\big(1-\sqrt{\lambda/\alpha(H)}\big)$-fraction of the players in that round. In each round i of these ℓ rounds, there are at most $\sqrt{\lambda/\alpha(H)}\cdot|V(i_0)|$ players in $V(i)$ for which round i is not edge-reducing. For a player $v \in V(i_0)$ to still have tasks at the end of the $2k$ rounds, at least k of these rounds are not edge-reducing for v (conditioned on the event that all the high probability events occur). The number of players for which this is the case can be at most $2k\sqrt{\lambda/\alpha(H)}\cdot|V(i_0)|/k$. Therefore, w.h.p,, in these $2k$ rounds at least a $\big(1-2\sqrt{\lambda/\alpha(H)}\big)$-fraction of the players in $V(i_0)$ are removed because all their tasks get assigned. Note that we assumed that $\lambda \leq \alpha(H)/5$ and therefore $2\sqrt{\lambda/\alpha(H)} \leq 2/\sqrt{5} < 1$. Consequently, after at most $2k\log(m)/\log(\sqrt{\alpha(H)/4\lambda}) = O(\log_{\alpha(H)/\lambda}(n)\cdot\log_{\alpha(H)/\lambda}(m)$ rounds that are edge-reducing for at least a $\big(1-\sqrt{\lambda/\alpha(H)}\big)$-fraction of the players, there are no players remaining and we are done. The number of rounds of the algorithm can therefore by upper bounded by

$$O\left(\frac{\log n}{\lambda} + \frac{\log m\log n}{\log^2(\alpha(H)/\lambda)}\right),$$

for any $\lambda \leq \alpha(H)/5$. For $\alpha(H) = O(1/\log m)$, choosing $\lambda = \alpha(H)/5$, the first term of the above expression dominates the second one, and we get a round complexity of $O\big(\log(n)/\alpha(H)\big)$. For large $\alpha(H)$ (i.e., if $\alpha(H) = \Omega(1/\log m)$), we set $\lambda = \log(\alpha\log m)/(\alpha\log m)$, and both terms evaluate to $O\big((\log m\log n)/\log^2(\alpha\log m)\big)$. □

Since each player writes a B-bit message on the blackboard in each round, the total bit complexity of the algorithm is $mB\cdot T$, where T is the running time from Theorem 6.6. In typical scenario where $m = O(n)$, this is optimal to within polylogarithmic factors (by Theorem 6.1). However, if the number of players greatly exceeds the number of tasks, it becomes wasteful to have all the players propose task assignments in each round.

7. ARBITRARY NETWORKS

Our results for the shared-blackboard model can be translated to the more decentralized setting, where players are connected by an arbitrary communication network, using the pipelining technique from [22]. Pipelining allows k pieces of information (henceforth called *tokens*) to be disseminated to all players in $D + k$ rounds, where D is the diameter of the communication network. The strategy is quite simple: each node keeps a pool of tokens it has received but not yet sent on, and in each round selects an arbitrary token from the pool and sends it. If the message size is large enough to allow $\beta > 1$ tokens per message, we can achieve better throughput by packing multiple tokens per message. Using an inductive argument that appears in [22] (also cf. [15]), the following statement can be shown:

LEMMA 7.1. *After $d + t$ rounds, every node v has either received all the tokens that originated at nodes at distance at most d from v, or node v has received at least $\beta\cdot t$ different tokens.*

In particular, at time $D + \lceil k/\beta\rceil$, each node has either received all tokens originating anywhere in the network, or it has received at least $\beta\cdot\lceil k/\beta\rceil \geq k$ different tokens; these amount to the same thing, since there are only k tokens in total. Consequently disseminating k tokens requires $O(D + k/\beta)$ rounds.

If the diameter D and the number of tokens k are known in advance, the strategy above allows all nodes to halt in $O(D+k/\beta)$ rounds; however, if D and k are not known, the nodes may not know when they have collected all tokens. It is easier to deal with an unknown number of tokens than with

an unknown diameter: if D is known but k is not, Lemma 7.1 shows that nodes may halt as soon as they reach a time $D+t$ in which fewer than $\beta \cdot t$ tokens have been received. Therefore we can still disseminate k tokens in $O(D+k/\beta)$ rounds. For the case where D is unknown, it is shown in [16] that w.h.p., an upper bound $\hat{D} \leq D + O(\sqrt{m/B} \cdot \log m)$ on D can be computed in time $\hat{D}$ (recall that m is the number of nodes in the network G). Thus we obtain:

LEMMA 7.2. *[16, 22] Even if D and k are not known to the nodes, with high probability, k tokens are disseminated (and all nodes can terminate) after $O(D + \sqrt{m/B} \log m + k/\beta)$ rounds.*

Distributed task allocation. We can now solve a given task allocation instance as follows. First, the players compute an upper bound of $\hat{D}$ on the diameter D, as described above. Then the players elect a unique leader among them (e.g., the player with the smallest ID). This can be done in $\hat{D}$ rounds. Afterwards, we start the token dissemination protocol from Lemma 7.2, using all the pairs $\{(v, x) \mid x \in X_v\}$ as our tokens. To avoid redundancy (and too many tokens), each player only forwards the first pair (v, x) that it receives for each task x; subsequent pairs (u, x) for $u \neq v$ are eliminated and not forwarded. It is not hard to show that this can be viewed as running the pipelining protocol from Lemma 7.2, except using tasks x as tokens, instead of pairs (v, x). At the end of the protocol, the leader node has received some pair (v, x) for each task x (or possibly more than one pair for some tasks); the leader now selects a permanent assignment for each task, and disseminates these assignments using the pipelining protocol.

THEOREM 7.3. TASKALLOCATION$_{m,n}$ *can be solved w.h.p. in time $O(D + \sqrt{m/B} \log m + n \log(n+m)/B)$.*

Remark. In the typical case, when $m = O(n)$, the time bound in the Theorem 7.3 simplifies to $O(D + n \log(n)/B)$. The algorithm described above is randomized, because the diameter-estimation algorithm from [16] is randomized. However, when $m = O(n)$ we can replace this part by a deterministic coarse upper bound on the diameter D: simply use pipelining to count the number of players in the network (see [15]), and use this number m as an upper bound on the diameter. This yields a deterministic $O(D + n \log(n)/B)$-round algorithm for task dissemination.

8. AN ALGORITHM FOR CONSTRUCTING ROOTED SPANNING TREES

We conclude our technical results with a simple algorithm for computing rooted spanning trees in directed broadcast networks. Our strategy is similar to the solution for task allocation in Section 7: we treat each network node as both a task and a player, where the input to player v is its set of in-children. Since we need to make sure that we do not create any cycles, we have to make task assignments in a more coordinated fashion than in Section 7. Therefore we assign children to parents in a top-down fashion, from the root towards the leaves.

We first assume that the nodes know the diameter D, or a linear upper bound on D. The first step, as in Section 7, is to select a leader r, which will serve as the root of the tree. Subsequently the algorithm runs in D phases. In the first phase, the root node r assigns all its in-neighbors as its children, and communicates this decision by applying the token dissemination protocol described in Lemma 7.2. In each subsequent phase we solve an instance of task allocation: the players are the nodes that are already assigned to some parent node and that still have unassigned in-neighbors; the tasks are all the unassigned in-neighbors of the players. Hence, the players of phase i are a subset of the nodes at in-distance $i-1$ from the root (the ones that have in-neighbors at in-distance i from the root), and the tasks are all the nodes at in-distance i from the root. The algorithm terminates as soon as all nodes are assigned to some parent node.

THEOREM 8.1. *The above algorithm solves the spanning tree problem in $O(D^2 + n \log(n)/B)$ rounds.*

PROOF. It follows from the construction of the algorithm that in phase i all nodes at in-distance i from the root are assigned to a parent node. Therefore, the time complexity of the algorithm is determined through D sequential executions of the task assignment protocol from Section 7. Let k_i be the number of nodes at in-distance i from the root. The number of tasks in tokens in phase i is k_i. Therefore the running time of the task assignment protocol of round i is $O(D + k_i)$ (recall that we assumed that the nodes know D). Hence, the overall time complexity is

$$O\left(D^2 + \sum_{i=1}^{D} k_i \cdot \frac{\log(n)}{B}\right) = O\left(D^2 + \frac{n \log n}{B}\right).$$

□

Dealing with an unknown diameter. If the diameter is initially unknown and we plug in the time complexity from Theorem 7.3, we obtain an overall time complexity of $O\big(D(D + \sqrt{n/B} \log n) + n \log(n)/B\big)$. This can be slightly improved by observing that phases do not necessarily need to be synchronized. As soon as a node receives a notification that it has been assigned to some parent node, it can start broadcasting its in-neighbors so that they can be assigned. With this modification, the additive $\sqrt{n/B} \log n$ penalty term for an unknown diameter is paid only once, instead of D times (once per phase); the penalty is dominated by the $n \log n/B$ in Theorem 8.1, so the overall time complexity from Theorem 8.1 is preserved. Note, however, that spanning tree constructed in this way is no longer a BFS tree.

9. DISCUSSION AND OPEN PROBLEMS

Our results in this paper leave several problems open. First, our lower bound from Section 5 shows that computing a spanning tree requires $\Omega(n/B)$ rounds, but the best upper bound of which we are aware, even assuming the diameter D is known in advance, is $O(\min\big\{D + |E|/B, D^2 + n \log n/B\big\})$. For dense networks with a large diameter, the bounds do not match. However, TASKALLOCATION$_{n,n}$ can be solved in $O(D + n \log n/B)$ rounds (see Section 7). The existence of a fast spanning tree algorithm implies a fast algorithm for task allocation, where we view each node as both a task and a player; the input of each player is its set of in-neighbors (viewed as "tasks"), and its output is the set of in-neighbors that chose it as their parent in the tree. The other direction is not necessarily true, since in general a task allocation

may contain cycles (when we view nodes as both tasks and players). If the network is sufficiently dense, and perhaps enjoys good expansion as well, is it nevertheless possible to use a fast task allocation algorithm to find a rooted spanning tree? Can we prove that cycles are unlikely to occur, and if so, can we resolve the few cycles that do occur quickly?

Another open problem concerns task-allocation with good task-player expansion and the hardness of finding a spanning tree in directed constant-degree expanders. In a constant-degree network with bounded bandwidth B, each node only receives $O(B)$ bits of information per round. This bottleneck bounds the number of nodes with which a given node can "exchange meaningful information", even though the diameter is small. To tackle this issue in a communication-complexity setting, we could charge the protocol not just for the total bits exchanged, but also for activating the (directed) channel between two players. We could then ask what is the smallest number of channels that must be activated to solve TaskAllocation or other problems. All the algorithms we have given for TaskAllocation require either all players to exchange information with all other players (as in the shared blackboard model), or one player to exchange $\Omega(n)$ information with all other players (as in the algorithm from Section 7). A strong lower bound on the number of player-to-player channels that must be activated would yield insight into the problem and perhaps lead to a lower bound on finding spanning trees in directed constant-degree expanders.

10. REFERENCES

[1] J. Aas. Understanding the Linux 2.6.8.1 CPU scheduler. Unpublished manuscript, 2005.

[2] B. Awerbuch, S. Kutten, and D. Peleg. Competitive distributed job scheduling. In *Proc. 24th Symp. on Theory of Computing (STOC)*, pages 571–580, 1992.

[3] Y. Bartal and A. Rosen. The distributed k-server problem–a competitive distributed translator for k-server algorithms. In *Proc. 33rd Symp. on Foundations of Computer Science (FOCS)*, pages 344–353, Oct 1992.

[4] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. In *Proc. 43rd Symp. on Theory of Computing (STOC)*, pages 363–372, 2011.

[5] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43:35:1–35:44, Oct. 2011.

[6] M. L. Dertouzos and A. K. Mok. Multiprocessor on-line scheduling of hard-real-time tasks. *IEEE Trans. Software Eng.*, 15(12):1497–1506, 1989.

[7] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-server algorithms. In *Journal of Computer and System Sciences*, pages 454–463, 1990.

[8] M. Frigo, C. E. Leiserson, and K. H. Randall. The implementation of the cilk-5 multithreaded language. In *Proc. 19th Conf. on Programming Language Design and Implementation (PLDI)*, pages 212–223, 1998.

[9] S. Frischknecht, S. Holzer, and R. Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proc. 22nd Symp. on Discrete Algorithms (SODA)*, pages 1150–1162, 2012.

[10] A. Gronemeier. Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness. In *Proc. 26th Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 505–516, 2009.

[11] T. S. Jayram. Hellinger strikes back: A note on the multi-party information complexity of AND. In *Proc. 13th Workshop on Randomization and Computation (RANDOM)*, pages 562–573, 2009.

[12] T. S. Jayram. Information complexity: a tutorial. In *Proc. 29th Symp. on Principles of Database Systems (PODS)*, pages 159–168, 2010.

[13] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990.

[14] E. Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009.

[15] F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Prof. 42nd Symp. on Theory of Computing (STOC)*, pages 513–522, 2010.

[16] F. Kuhn and R. Oshman. The complexity of data aggregation in directed networks. In *Proc. of 25th Symp. on Distributed Computing (DISC)*, pages 416–431, 2011.

[17] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[18] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.

[19] B. Patt-Shamir. A note on efficient aggregate queries in sensor networks. *Theor. Comput. Sci.*, 370(1-3):254–264, 2007.

[20] A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106:385–390, December 1992.

[21] A. A. Shvartsman and C. Georgiou. *Cooperative Task-Oriented Computing: Algorithms and Complexity*. Morgan&Claypool Publishers, 2011.

[22] D. M. Topkis. Concurrent broadcast for information dissemination. *IEEE Trans. Softw. Eng.*, 11:1107–1112, October 1985.