Distributed Resource Discovery in Sub-Logarithmic Time

Bernhard Haeupler[*]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
haeupler@cs.cmu.edu

Dahlia Malkhi
VMware Research
Palo Alto, CA 94304, USA
dmalkhi@vmware.com

ABSTRACT

We present a new distributed algorithm for the resource discovery problem introduced by Harchol-Balter, Leighton, and Levin in PODC'99.

The resource discovery problem consists of a synchronous network with n machines in which at any timestep any machine v can PUSH or PULL a message to/from any other machine u whose (IP) address is known to v. Messages can contain addresses which then change the "topology". The goal of a distributed resource discovery problem is to enable all machines to learn the addresses of all other machines as fast as possible while keeping the number of messages sent low.

We present a randomized distributed algorithm that achieves this goal in $O(\log D \log\log n)$ rounds using $O(n)$ messages, where D is the strong diameter of the initial topology. Up to the $\log\log n$ factor, our round complexity is best possible given the trivial $\Omega(\log D)$ lower bound. For many typical networks with $D = o(n)$, our running time is a drastic improvement over prior $O(\log n)$ round algorithms. In particular, for networks with polylogarithmic diameter an $O(\log^2 \log n)$ running time and thus an almost exponential speedup is obtained.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY—*Nonnumerical Algorithms and Problems*; F.1.2 [**Theory of Computation**]: COMPUTATION BY ABSTRACT DEVICES—*Modes of Computation* Parallelism and concurrency

General Terms

Algorithms, Theory, Performance, Reliability

[*]Research supported in part by NSF award "AF:Small:Distributed Algorithms for Near-Planar Networks".

Keywords

resource discovery; direct addressing; gossip; information dissemination; rumor spreading

1. INTRODUCTION

We give a new distributed algorithm for the classical *resource discovery* problem introduced in [11] that features drastically improved running time on typical networks while maintaining an optimal message complexity.

Resource discovery is a basic task in distributed networks that is concerned with informing all participants in a network about each other in a fast and communication efficient way. It can be used as a coordination mechanism, e.g., in networks with somewhat frequent joins and leaves, and often forms the first step for implementing further distributed functionalities. As such, it has been intensely studied from both a practical and theoretical perspective.

1.1 Formal Problem Definition

The mathematical definition as a distributed problem goes back to the work of Harchol-Baler et al. [11] and arose in the context of building a large-scale distributed cache for their newly founded company Akamai:

Machines are abstracted as nodes in a directed graph $G = (V, E)$ with n nodes in which a directed edge $(u, v) \in E$ means that machine u knows (the IP address of) v and can thus contact v. We write $\Gamma(u)$ to denote set of nodes that node u can contact. Communication proceeds in synchronous rounds in which each node u can PUSH or PULL a message to/from any nodes in $\Gamma(u)$. Each such operation gets counted as one message. In a message a node can communicate any desired information but typically a node simply sends the set of IDs it knows and possibly some small amount of control information, e.g., $O(\log n)$ bits. Once a node has received a message containing addresses it can contact the respective nodes in subsequent rounds and the network graph G changes accordingly. This network model has also been called the DIRECT ADDRESSING or DA model [8].

The goal of distributed resource discovery algorithms in the DA model is to guarantee that for a given initial network topology all nodes learn about each other as quickly as possible without sending too many messages.

1.2 Prior Work on Resource Discovery

The trivial resource discovery algorithm in the DA model is flooding. Flooding simply has every node forward its knowledge to *all* nodes it knows about. It is easy to see that this flooding algorithm terminates in $\log D + 1$ time where D is the (weak) diameter[1] of the initial network topology and that this running time bound is optimal. On the other hand, the flooding algorithm has a horrible message complexity and sends up to $\Theta(n^2)$ messages in a single round. This is even true for sparse networks as they become dense quickly.

The idea of Harchol-Balter et al. was to leverage randomization in achieving more communication efficient information dissemination. They introduced the NameDropper gossip algorithm in which each node sends the addresses it knows solely to a uniformly random contact. They proved that $O(\log D \log n)$ rounds each using n messages are sufficient for this algorithm to solve the resource discovery problem. Subsequently Law and Siu [18] gave a simple randomized resource discovery algorithm requiring $O(\log n)$ rounds and Kutten, Peleg, and Vishkin [17] building on a connectivity algorithm of Shiloach and Vishkin [20] gave a deterministic algorithm that uses $O(\log n)$ rounds and sends only $O(n \log n)$ messages in total.

1.3 Our Result

While these $O(\log n)$ running times match the trivial $\Omega(\log D)$ lower bound for networks with a linear (or polynomial) diameter, such networks are arguably not the setting of interest. Indeed the networks topologies one would expect to encounter in a resource discovery setting are typically well connected and have a small, e.g., at most (poly-)logarithmic, strong diameter[2]. In those networks, the $\Omega(\log D)$ lower bound merely excludes the existence of sub-log-logarithmic algorithm thus leaving open the question whether such exponentially faster algorithms are achievable. In this work we answer this question in the affirmative by giving an algorithm that matches the $\Omega(\log D)$ lower bound for the strong diameter up to an $O(\log \log n)$ factor:

THEOREM 1. *Suppose a network in the DA model with n nodes and a strong diameter of D. There is a distributed randomized algorithm that, with high probability, completes a resource discovery in any such network in $O(\log D \cdot \log \log n)$ rounds using a total of $\Theta(n)$ messages.*

Remarks:

- Like all previous works we do not attempt to give any formal failure model and prove robustness results for our algorithm with respect to any permanent or temporary failure or interleaved join and leaves. This said, we believe that the randomized GOSSIP nature of our algorithm (similar to [11]) actually makes the algorithm quite robust at least to random failures or leaves (which, e.g., are unlikely to specifically delete cluster-centers). More over the (almost) exponentially faster running time guarantee on typical networks compared to prior approaches drastically reduces the timeframe in which the network needs to be (sufficiently) stable and allows the rebuilding process to be run more frequently.
- Like [18] but in contrast to [11, 17] we give running time guarantees with respect to the strong diameter. We postulate this diameter to be small for instances of interest. We also remark that it is generally impossible to have GOSSIP algorithms that perform well in graphs with low weak-diameter while having nodes send at most c messages per round. In fact such algorithms cannot run in less than $\log_c n = \log n / \log c$ rounds on the star-network in which one center node knows about all other n nodes (but not vice versa) even though this network has a weak diameter of 2. Thus the only way to even achieve sub-logarithmic running time in this network is to have a single node send at least polynomially many messages in one round.

1.4 More Prior Work

In addition to the results of [11, 17, 18] on the resource discovery problem (see Section 1.2) the techniques used in our algorithm are also closely related to [8, 2]. These works show that one can achieve sub-logarithmic running times for the global broadcast problem if one considers a communication model that allows for DIRECT ADDRESSING and also for contacting a random participating node in each round. In particular, Avin and Elsässer[2] gave an algorithm with $O(\sqrt{\log n})$ round complexity while Haeupler and Malkhi [9] showed that even a $\Theta(\log \log n)$ running time is achievable and optimal. These results can be interpreted as running times of resource discovery algorithms on random topologies in which each node has many directed links to independently chosen nodes. Since such topologies have a logarithmic diameter Theorem 1 implies an $O(\log^2 \log n)$ running time for this setting as well. The setting in this paper however is significantly more involved because no randomness or even expansion in the topology is assumed. The one (and only) crucial idea we managed to carry over from [8] is to organize nodes in clusters and try to achieve a doubly exponential growing process by having clusters of size s reach out to $O(s)$ clusters for merging thus leading to s^2 size clusters in one step.

We also remark that there are many parallels between this work and gossip algorithms for the LOCAL model [19], which features a fixed undirected topology. There too the resource discovery or global broadcast problem can be solved via flooding in an optimal time complexity, which is $O(D)$ in the LOCAL model. This however requires up to $\Theta(n^2)$ messages per round. Demers et al. [6] were the first to introduce randomized gossip algorithm to achieve fast and reliable information dissemination. Since then uniform gossip has been shown to work well on complete graphs [12] and random and expanding topologies [4, 7]. Similar to this work more recent non-uniform gossip algorithms [3, 8] even achieve $O(D + \text{poly} \log n)$ running times, which are close to the trivial $\Omega(D)$ lower bound achieved by flooding albeit

[1]The weak diameter of a network is the smallest integer for which every node has a path of length at most D to any other node when directions of edges are ignored.

[2]The strong diameter of a network is the smallest integer for which every node has a directed path of length at most D to any other node.

while using only $O(1)$ communication (instead of $O(n)$) per node.

Beyond these works, owing to the high practical relevance of the topic, many works have studied variants of the resource discovery problem. We mention [15, 1, 16] which consider resource discovery in an asynchronous setting, [10, 13] which considers communication efficient resource discovery gossip with small messages and refer to [14] and the recent PhD thesis of Davtya [5] for a broader overview and further references.

2. RESOURCE DISCOVERY

2.1 Clusters

In this section we introduce clusters, a simple tool used in this paper to achieve efficient exploration. Briefly, a cluster is a set of nodes that have a designated cluster-center known to all nodes in the cluster. All nodes within a cluster can exchange information via the center using only a constant number of rounds and messages per node. This allows a cluster to act in a coordinated manner.

More precisely, a cluster is a set S of nodes, with each node $v \in S$ containing a variable $follow_v = \ell_S$ set to the address of the cluster leader. Initially all nodes form their own cluster and have $follow$ set to their own address. For a node v, we denote its cluster by $C(v)$ (or simply C when clear from context).

Having a central leader which is known to all nodes in a cluster allows a cluster to coordinate using only a constant number of communication rounds. In particular, in one communication round followers may PUSH information to the leader and thus have the leader collect inputs from the entire cluster. In a second round, followers may PULL a response from the leader. Throughout the rest of this paper we implicitly assume that information is exchanged within each cluster before external communication steps as needed. This allows us to treat clusters as indivisible entities which act as a unit and share full information, while omitting the details concerning PUSHing and PULLing messages.

We furthermore use the following notation: For two clusters C_1, C_2, if there is an edge from a node $v \in C_1$ to a node $w \in C_2$ we say that C_2 is a *cneighbor* of C_1. Clusters generally do not know which clusters are cneighbors. For this reason any cluster C maintains a set $A(C)$ in which it keeps track of clusters that it knows to be current cneighbors.

Remark:
A cluster can easily find out, using two cluster internal PUSH/PULL rounds, which nodes $\Gamma(C)$ the nodes in C are connected to. However, it is much harder (or message intensive) to find out what its distinct cneighbors are, i.e., which nodes in $\Gamma(C)$ belong to the same cluster. In particular, as clusters merge these clusterings change and it is generally too message expensive to keep a complete list of cneighbors updated. This is the reason for having $A(C)$. It also means that after every round of cluster merging the list $A(C)$ becomes outdated and is thus set to be the empty set.

2.2 The Resource Discovery Algorithm

We next give a high level description of our algorithm:

The algorithm first calls the INITIALIZATION sub-routine which reduces the number of clusters by a polylogarithmic factor, bringing the average cluster size to $s = \log^{O(1)} n$. This is done in $\Theta(\log\log n)$ rounds via a merging process that guarantees singly exponential reduction process in the number of clusters.

The main loop of the algorithm then employs iterations with a doubly exponential growth progress by roughly squaring the average size s of a cluster in each iteration. For this, each iteration begins with using the DISCOVERCNEIGHBORS routine to have clusters explore their neighborhood and discover at least $d \approx s$ cneighbors as potential merging partners.

Clusters that do not have enough cneighbors use INFLATEOUTDEGREE to perform a local flooding procedure until at least d clusters are reached. This requires at most $\log D$ flooding steps and does not lead to a large message overhead because only clusters with less then d cneighbors participate.

The SAMPLEMERGE routine then randomly sub-samples roughly a $1/d \approx 1/s$ fraction of clusters to be the new cluster-centers and assigns each non-selected cluster to a cluster center. With high probablity, the number of clusters is thus reduced roughly by factor s, which leads to a new average cluster size of about $s \cdot d \approx s^2$.

This doubly exponential merging process is performed for $O(\log\log n)$ iterations until the average cluster size reaches at least $\sqrt{n}\log n$. A final INFLATEOUTDEGREE guarantees that every cluster becomes aware of any other cluster. The cluster centers then inform all their nodes about all other nodes, which completes the resource discovery process.

Algorithm 1 Log-logarithmic resource discovery algorithm

```
1: INITIALIZATION($\Theta(\log\log n)$)
2: $\epsilon \leftarrow 1/2$
3: $s \leftarrow \log^{2/\epsilon} n$
4: Clusters of size $s' > s$ split into $\lceil s'/s \rceil$ clusters of size
   between $s/2$ and $s$

5: repeat
   ▷ explore/expand neighbors to discover $s^{1-\epsilon}$ cneighbors
6:     DISCOVERCNEIGHBORS($s, 2/\epsilon$)
7:     $d \leftarrow s^{1-\epsilon}$
8:     INFLATEOUTDEGREE($d$)

       ▷ sample-merge phase to reduce number of clusters by
   roughly $1/d$
9:     SAMPLEMERGE($s, d$)
10:    $s \leftarrow \Theta(s \cdot d/\log n)$     ▷ $\Theta(s \cdot d/\log n) = \Omega(s^{2-1.5\epsilon})$
11:    Clusters of size $s' > s$ split into $\lceil s'/s \rceil$ clusters of size
   between $s/2$ and $s$

12: until $s \geq \sqrt{n}\log n$               ▷ $\Theta(\log\log n)$ repetitions

13: INFLATEOUTDEGREE($\sqrt{n}$)   ▷ all clusters become aware
    of all clusters
14: Cluster centers inform all nodes in their cluster about
    all other nodes
```

2.3 Analysis

Next we prove our main theorem:

THEOREM 1 (repeated). *Suppose a network in the DA model with n nodes and a strong diameter of D. There is a distributed randomized algorithm that, with high probability, completes a resource discovery in any such network in $O(\log D \cdot \log\log n)$ rounds using a total of $\Theta(n)$ messages.*

The proof relies on the following guarantees about the subroutines referenced in Algorithm 1. We give their statements here and provide their implementations and proofs in subsequent subsections.

LEMMA 2. *Assume that there are at least $\Omega(\log n)$ clusters before a call to* INITIALIZATION. *For every r, with high probability,* INITIALIZATION*(r) reduces the number of clusters by a factor of $c = e^{\Theta(r)}$ using $O(r)$ rounds and $\Theta(n)$ messages in total.*

LEMMA 3. *Assume each cluster is of size at most s and suppose that $s \geq \log^2 n$. With high probability after a call of* DISCOVERCNEIGHBORS*(s, r) each cluster C knows either all or at least $|A(C)| \geq \Theta(s^{1-(2/r)})$ distinct cneighbors. The routine furthermore requires at most $O(r)$ rounds with at most $O(rs^{1-(1/r)})$ messages being sent by each cluster.*

LEMMA 4. *Assume that each cluster has outgoing edges to either to all of its cneighbors or to at least d of them. Assume further that the underlying network has a strong diameter of at most D. Then, at the end of* INFLATEOUTDEGREE*(d) a cluster C is aware of either all clusters or at least d cneighbors, i.e, $|A(C)| \geq d$. Furthermore, only $O(\log D)$ rounds and $O(d \cdot \log D)$ messages per cluster are required.*

LEMMA 5. *Assuming that at the beginning of* SAMPLEMERGE*(d) every cluster C is aware of at least $d \geq \log^2 n$ cneighbors, i.e., $|A(C)| \geq d$, then with high probability* SAMPLEMERGE*(d) reduces the number of clusters by a factor of $\Theta(\log n/d)$. Furthermore, only $O(1)$ rounds and $O(d)$ messages per cluster are required.*

PROOF OF THEOREM 1. We first prove the correctness of Algorithm 1. Initially each node forms its own cluster so there are n clusters. Theorem 2 now guarantees that the initialization step ends with at most $n/\log^{2/\epsilon} n = n/s$ clusters. The splitting of clusters of size larger than s leads to at most n/s further clusters for a total of at most $2n/s$ clusters each of size at most s.

We prove by induction that this guarantee of at most $2n/s$ clusters of size at most s holds whenever the beginning or end of the main loop is reached.

In particular, Theorem 3 guarantees for all $eps > 0$, that during DISCOVERCNEIGHBORS$(s, 2/\epsilon)$ every cluster discovers at least $d = s^{1-\epsilon}$ or all its cneighbors and Theorem 4 guarantees that after INFLATEOUTDEGREE(d) every node has at least d cneighbors in its $A(C)$ set. Theorem 5 shows that SAMPLEMERGE(d) then reduces the number of clusters by a $\Theta(\log n/d)$ factor thus decreasing the number of clusters to at most $\Theta(n/(\frac{s \cdot d}{\log n}))$ or n/s after the parameter s is updated. Splitting clusters larger than s again at most doubles the number of clusters to $2n/s$ and leads to clusters of size at most s as postulated.

Once the main loop terminates this leads to at most $2\sqrt{n}/\log n$ clusters. A final call to INFLATEOUTDEGREE$(\sqrt{n})$, according to Theorem 4, then leads to all clusters being aware of all other clusters which allows the cluster centers to inform all nodes in their cluster about all other nodes. The algorithm thus terminates correctly.

Next we analyze the number rounds and number of messages required for this process. Theorem 2 guarantees that the initialization takes $\Theta(\log\log n)$ rounds and $\Theta(n)$ messages. For the main loop we claim that there are at most $4\log\log n$ iterations of the main loop of Algorithm 1 and that each such iteration uses only $O(\log D)$ rounds and at most $O(n/\log n)$ messages. The number of iterations follows simply from the observation that each iteration increases s to $\Theta(s^{2-\epsilon}/\log n) \geq \Theta(s^{2-\epsilon}/s^{\epsilon/2}) = \Theta(s^{2-1.5\epsilon})$. Even for $\epsilon = 1/2$ having $i = 4\log\log n$ iterations s would increase s from $\log^2 n$ to $(\log^2 n)^{1.25^i} \geq 2^{2^{\log\log n}} = n$ which implies that the loop terminates after less than $4\log\log n$ iterations. Regarding the number of messages and rounds, Theorem 3 states that DISCOVERCNEIGHBORS$(s,2/\epsilon)$ uses at most $O(2/\epsilon s^{1-\epsilon/2})$ messages for each of the $2n/s$ clusters for a total of at most $O(n/s^{\epsilon/2}) = O(n/\log n)$ messages per iteration. It also requires only $O(1)$ rounds. Similarly, SAMPLEMERGE(d) uses a total of $2n/s \cdot ds = O(n/s^{\epsilon}) = O(n/\log^2 n)$ messages and $O(1)$ rounds.
Lastly, INFLATEOUTDEGREE(d) uses

$$2n/s \cdot d\log D = O(n\log n/s^{\epsilon}) = O(n/\log n)$$

messages and $\log D$ rounds. This shows that the main loop requires overall at most $O(\log D \log\log n)$ rounds and $o(N)$ messages.

To complete the analysis, note that the final call of INFLATEOUTDEGREE$(\sqrt{n})$ also requires at most $O(\log D)$ rounds and at most $O(\log D\sqrt{n}) \cdot 2n/(\sqrt{n}\log n) = O(n)$ messages. □

2.4 Initialization

The initialization sub-routine brings down the number of clusters in a network by a polylogartihmic factor by having $O(\log\log n)$ rounds in which a constant fraction of the clusters merges.

In a round, the goal is to break symmetry into joiners and joinees, such that sufficiently many joiners merge with joinees. In a full topology, it would suffice to flip a coin to become joiner/joinee and then select a merging partner at random; this would succeed for a constant fraction of the network. In general topologies this simple rule does not work, because some nodes may be more important than others due to the graph structure. For example, in a star graph, the center determines the success/failure of all merging attempts. To address this, we first use a randomized neighbor-selection step which identifies important nodes. After this step, every node which received more than one neighbor-request chooses to be a joinee and immediately accept any joiner request. There will remain a group of nodes which are neither joinees, nor have their requests accepted already. The nice thing is that each one of these nodes will be matched with at most one incoming neighbor request and one outgoing request. Therefore, within this set, flipping a coin to become joiner/joinee will succeed in merging a constant fraction with high probability.

Algorithm 2 INITIALIZATION(r)

```
1: for $r$ iterations do
2:     every cluster $C$ picks one neighbor $v \in \Gamma(C)$ and sends a merge request $C(v)$
3:     if cluster received more than one merge request then
4:         retract your own merge request    ▷ choose to become joinee
5:         send all (non-retracted) incoming requests a joinee signal
6:         accept any (non-retracted) incoming requests
7:     end if
8:     with probability 1/2 retract your own merge request
9:     if did not receive a joinee signal then
10:        accept any (non-retracted) incoming request
11:    end if
12:    retract any merge requests that have not been accepted yet
13: end for
```

LEMMA 2. *Assume that there are at least $\Omega(\log n)$ clusters before a call to* INITIALIZATION. *For every r, with high probability,* INITIALIZATION*(r) reduces the number of clusters by a factor of $c = e^{\Theta(r)}$ using $O(r)$ rounds and $\Theta(n)$ messages in total.*

PROOF. Suppose that the number of clusters before any of the r iterations is $N = \Omega(\log n)$. We prove that the iteration reduces the number of clusters by a constant factor with high probability. For this we consider the directed request graph which indicates which cluster requested to be merged with which other cluster. The merge requests guarantee that every node has an out-degree of exactly one. Since there are exactly N requests the number of nodes that retract their request because of multiple received requests is at most $N/2$. We want to show that at least a constant fraction of these requests are accepted. Some are already accepted at Line 5 by the nodes that retracted their requests. If these make up $N/8$ requests the claim is proved. If not then there are at least $N/4$ remaining requests.

Since we resolved requests to nodes with an in-degree larger than one we have the guarantee that these remaining requests decompose into directed cycles and directed paths. In these paths and cycles we break symmetries using randomization. In particular, any remaining edge has a probability of 1/4 of being accepted because with probability 1/2 it is not retracted by its requester and with an independent probability of 1/2 it is accepted by the cluster that received the request. Furthermore, due to the paths structure the $N/4$ remaining requests contain at least $N/8$ requests that are accepted independently. Out of these $N/8$ requests we expect $N/32$ to be accepted and a standard Chernoff bound shows that with high probability at least $\Theta(N)$ will be accepted. This proves that every iteration leads with high probability to at least a constant fraction of the clusters merging. Doing this for r rounds leads to the claimed $e^{\Theta(r)}$ overall reduction in the number of clusters.

To determine the message complexity we note that each cluster sends only $O(1)$ messages per iteration. This leads to at most $O(n)$ messages in the first iteration. Since the number of clusters in subsequent iterations decreases geometrically with a factor of $\delta < 1$ the overall message complexity is also $\sum_{i=0}^{r} \delta^i O(n) = O(n)$. □

2.5 Cneighbor Discovery

The cneighbor-discovery sub-routine probes the nodes neighboring a cluster in order to identify sufficiently many distinct cneighbors. At the beginning of sub-routine, a cluster C has a set $\Gamma(C)$ of neighboring nodes which are split, potentially unevenly, among an unknown set of cneighbors. In a round, the cluster probes roughly s neighbors. It stores all the cneighbors it discovered in $A(C)$ and repeats with remaining neighbors. Intuitively, each such round makes progress in one of two ways. One case is that it discovers a large enough number of 'light' cneighbors, which do not contain a lot of neighbors, and we are done. The other case is that it discovers 'heavy' cneighbors, which contain a total large number of neighboring nodes. In this case, in the next round, C has a significantly reduced set of neighbors to work on.

The algorithm is given as Algorithm 3.

Algorithm 3 DISCOVERCNEIGHBORS(s, r)

```
1: $X \leftarrow \Gamma(C)$
2: $A(C) = \emptyset$
3: for $2r$ repetitions do
4:     select a random set $R \subseteq X$ of size $\min\{s^{1-(1/r)}, |X|\}$
5:     PULL from each $v \in R$ the members of $C(v)$
6:     $A(C) \leftarrow A(C) \cup \left(\bigcup_{v \in R}\{C(v)\}\right)$
7:     $X \leftarrow X \setminus \left(\bigcup_{v \in R} C(v)\right)$
8: end for
```

LEMMA 3. *Assume each cluster is of size at most s and suppose that $s \geq \log^2 n$. With high probability after a call of* DISCOVERCNEIGHBORS*(s, r) each cluster C knows either all or at least $|A(C)| \geq \Theta(s^{1-(2/r)})$ distinct cneighbors. The routine furthermore requires at most $O(r)$ rounds with at most $O(rs^{1-(1/r)})$ messages being sent by each cluster.*

PROOF. Define $t = s^{1-(1/r)}$. Consider one cluster C and denote $C_1, \ldots, C_m$ its cneighbors. We define the degree $\Delta_C(C_i)$ of a cneighbor C_i to be the number of edges going from C to distinct nodes in C_i. We classify these cneighbors according to this degree into two groups. The first group contains all "heavy" cneighbors whose degree makes up at least a $c\log(n)/t$ fraction of X for some constant $c > 1$, i.e., $\Delta_C(C_i) \geq c|X|\log(n)/t$, while the second group contains all other "light" cneighbors of C.

If at any iteration the size $|X|$ of the remaining neighborhood becomes smaller than t then all neighbors are contacted completing the claim. The remainder of this proof is concerned with the case that $|X|$ remains large and t neighbors get selected in each iteration.

Since the t neighbors in R are chosen independently at random each heavy cneighbor is PULLED in expectation $c\log n$ times and with high probability at least once. In particular the probability of not being PULLed at all is at most $(1 - \frac{c|X|\log(n)/t}{|X|})^t < e^{-c\log n}$.

Conversely, every light cneighbor is hit with at most an expected $c\log n$ PULLs from C and a standard Chernoff bound shows that with high probability at most $O(\log n)$ PULLs go to any such cneighbor.

We consider two cases. If the sum total of degrees going into light cneighbors is at least a $1/t^{(1/r)}$ fraction then the number of edges selected from this set is in expectation and with high probability $\Omega(t^{(1-(1/r))}) = \Omega(s^{(1-(1/r))(1-(1/r))}) = \Omega(t^{(1-(2/r))})$. The number of distinct light cneighbors hit is therefore at least $\Omega(t^{(1-(2/r))})/O(\log n)$ with high probability finishing the claim.

The other possibility is that the total number of edges going into light cneighbors is less than a $1/t^{(1/r)}$ fraction. Since these are the only edges that make up the total degree $|X|$ the new set X' of the next iteration has size $|X'| \leq |X|/t^{(1/r)}$.

Thus each iteration with high probability either discovers enough distinct cneighbors or reduces the total degree by a $t^{(1/r)}$ factor. Furthermore, if the algorithm does succeed in the first round we know that in this iteration at least one cluster of degree $c|X|\log n/t$ existed. Since the degree can be at most the size of the cluster which is assumed to be at most s we get that either the first iteration succeeds or $|X| \leq st/c\log n \leq s^2$. Therefore, failed iterations which reduce $|X|$ by a factor of $1/t^{(1/r)}$ can happen at most $2r$ times before $|X| \leq t$ and all remaining cneighbors are contacted. □

2.6 Degree Inflation

The sub-routine for degree-inflation works to expand the out-degree of all clusters to a desired threshold d. In a round, every 'lean' node, whose degree has not reached the desired threshold, simply PULLs from all of its neighbors their neighbor-sets. This means that a cluster PULLs information about nodes at twice the distance of the previous round. Therefore, in at most $\log D$ rounds, the entire graph can be PULLed.

Algorithm 4 InflateOutdegree(d)

```
1: for $\log D$ iterations do
2:     if $|A(C)| < d$ then
3:         pull $A(C')$ from every $C' \in A(C)$ into $A(C)$
4:     end if
5: end for
```

Lemma 4. *Assume that each cluster has outgoing edges to either to all of its cneighbors or to at least d of them. Assume further that the underlying network has a strong diameter of at most D. Then, at the end of* InflateOutdegree*(d) a cluster C is aware of either all clusters or at least d cneighbors, i.e,* $|A(C)| \geq d$*. Furthermore, only* $O(\log D)$ *rounds and* $O(d \cdot \log D)$ *messages per cluster are required.*

Proof. For analysis purposes, we fix a distance function $d()$ reflecting the shortest cluster-to-cluster hop length between clusters at the beginning of the procedure. Clearly, $d(C_1, C_2) \leq D$ for every two clusters C_1, C_2. InflateDegree progresses in iterations. At the start of each iteration, a cluster C is called *lean* if $|A(C)| < d$. Clusters continue iterating the loop only as long as they remain lean.

We prove the lemma by induction on the following invariant: For every lean cluster C entering its k'th loop iteration, $A(C)$ contains all clusters W with $d(C,W) \leq 2^{(k-1)}$. The base of the induction for $k = 1$ holds since if C is initially lean then it has outgoing edges to all of its distance-1 cneighbors. Suppose C is lean at the beginning of iteration $k+1$. Consider any cluster W with $D(C,W) \leq 2^k$. There exists some cluster W' on the path from C to W, such that $D(C,W') \leq 2^{(k-1)}$ and $D(W',W) \leq 2^{(k-1)}$. Let's examine $A(C)$ and $A(W')$ at the k'th iteration. When the iteration starts, we know that C is lean, and by the induction hypothesis $W' \in A(C)$. We know that C remains lean at the end of the k'th iteration after it pulls $A(W')$ into $A(C)$. Therefore, when the k'th iteration starts, W' too is lean, and by the induction hypothesis, $W \in A(W')$. Therefore, W is pulled into $A(C)$ at the k'th iteration. This completes the induction.

Clearly, within $\log(D)$ iterations, C learns about all existing clusters and we are done. Each cluster furthermore only uses less than d connections per iterations for a total of at most $d \cdot \min\{\log d, \log D\}$ connections per cluster. □

2.7 Sample and Merge

A sample-and-merge sub-routine starts with at most N/s clusters all having d out-cneighbors. Then, a fraction of roughly $1/d$ are sampled to become center-clusters (more precisely, we sample with probability $c\log n/d$, to guarantee that every cluster has a cneighbor which is sampled). Each non-center cluster selects one sampled cluster among its d cneighbors to join. At the end of the sub-routine, the number of clusters is reduced by a factor of roughly $d/(c\log n)$).

Algorithm 5 SampleMerge(d)

```
1: with probability $c\log n/d$ become a cluster-center
2: if non-center cluster then
3:     contact $d$ clusters from $A(C)$ to find a center-cluster
4:     merge with an arbitrary neighboring center-cluster
5: end if
```

Lemma 5. *Assuming that at the beginning of* SampleMerge*(d) every cluster C is aware of at least* $d \geq \log^2 n$ *cneighbors, i.e.,* $|A(C)| \geq d$*, then with high probability* SampleMerge*(d) reduces the number of clusters by a factor of* $\Theta(\log n/d)$*. Furthermore, only* $O(1)$ *rounds and* $O(d)$ *messages per cluster are required.*

Proof. Let x be the number of clusters before SampleMerge. After sampling, the expected number of center-clusters is $\mu = x \cdot (c\log n/d)$. Since $x \geq |A(C_1)| = d$ we have $\mu \geq \log n$. Since each cluster makes an independent decision, a standard Chernoff bound shows that with high probability there are exactly $\Theta(\mu)$ center-clusters which form the new clusters after SampleMerge. A similar argument guarantees that with high probability out of the d neighbors contacted by a non-center cluster at least one center-cluster is found. This guarantees that no non-center clusters survive the SampleMerge procedure. □

3. CONCLUSION AND OPEN PROBLEMS

The work presented in this paper makes a large improvement in the round complexity of resource discovery on instances of interest while maintaining the overall communication optimal. There are a number of compelling problems left to tackle.

First, while the message complexity of our algorithm is $O(n)$ there are nodes that send more than $O(1)$ messages per round. We believe that it is possible to design a gossip algorithm with the same running time guarantees in which nodes send at most one message per round. This requires a modified INITIALIZATION routine which does not just guarantee that the number of clusters is reduced by a polylogarithmic factor but instead guarantees that *each* cluster is of polylogarithmic size. More importantly the SAMPLEMERGE routine needs to be followed by an intricate re-balancing step which guarantees that each cluster-center is joined by sufficiently many clusters. We defer these details to the journal version.

Second, our work leaves open to further improve the network discovery time of graphs with only a weak diameter bound. Our intuition is that a crucial building block for solving this would be a log-logarithmic local-flooding algorithm. Once achieved, it may be repeated $\log D$ times to fold the entire graph into a clique, in a similar manner to the utilization of local broadcast in the gossip techniques of [3].

4. REFERENCES

[1] I. Abraham and D. Dolev. Asynchronous resource discovery. *Computer Networks*, 50(10):1616–1629, 2006.

[2] C. Avin and R. Elsässer. Faster Rumor Spreading: Breaking the $\log n$ Barrier. In *Proceedings of the International Symposium on Distributed Computing*, DISC '13, pages 209–223, 2013.

[3] K. Censor-Hillel, B. Haeupler, J. Kelner, and P. Maymounkov. Global Computation in a Poorly Connected World: Fast Rumor Spreading with no Dependence on Conductance. In *Proceedings of the ACM Symposium on Theory of Computing*, STOC '12, pages 961–970, 2012.

[4] F. Chierichetti, S. Lattanzi, and A. Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 399–408, 2010.

[5] S. Davtyan. *Resource Discovery and Cooperation in Decentralized Systems.* PhD thesis, University of Connecticut, 2014.

[6] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, 1987.

[7] G. Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 57–68, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[8] B. Haeupler. Simple, fast and deterministic gossip and rumor spreading. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 705–716, 2013.

[9] B. Haeupler and D. Malkhi. Optimal gossip with direct addressing. In *Proceedings of the 31st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 176–185, 2014.

[10] B. Haeupler, G. Pandurangan, D. Peleg, R. Rajaraman, and Z. Sun. Discovery through gossip. In *Proceedinbgs of the 24th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 140–149, New York, NY, USA, 2012. ACM.

[11] M. Harchol-Balter, T. Leighton, and D. Lewin. Resource discovery in distributed networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC'99, pages 229–237, 1999.

[12] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 565–574, Washington, DC, USA, 2000. IEEE Computer Society.

[13] S. Kniesburges, A. Koutsopoulos, and C. Scheideler. A deterministic worst-case message complexity optimal solution for resource discovery. *Theoretical Computer Science*, 2014.

[14] K. M. Konwar, D. Kowalski, and A. A. Shvartsman. Node discovery in networks. *Journal of Parallel and Distributed Computing*, 69(4):337–348, 2009.

[15] S. Kutten and D. Peleg. Asynchronous resource discovery in peer to peer networks. In *Reliable Distributed Systems, 2002. Proceedings. 21st IEEE Symposium on*, pages 224–231. IEEE, 2002.

[16] S. Kutten and D. Peleg. Asynchronous resource discovery in peer-to-peer networks. *Computer Networks*, 51(1):190–206, 2007.

[17] S. Kutten, D. Peleg, and U. Vishkin. Deterministic resource discovery in distributed networks. *Theory of Computing Systems*, 36(5):479–495, 2003.

[18] C. Law and K.-Y. Siu. An $O(\log n)$ randomized resource discovery algorithm. In *Brief Announcements of the 14th International Symposium on Distributed Computing, Technical University of Madrid, Technical Report FIM/110.1/DLSIIS*, pages 5–8. Citeseer, 2000.

[19] D. Peleg. *Distributed computing: a locality-sensitive approach.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

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