Time-Optimal Information Exchange on Multiple Channels

Stephan Holzer[1], Yvonne-Anne Pignolet[2] *, Jasmin Smula[1], Roger Wattenhofer[1]
[1]Computer Eng. and Networks Laboratory (TIK), ETH Zurich, Switzerland
[2]ABB Corporate Research, Dättwil,Switzerland
{stholzer,smulaj,wattenhofer}@tik.ee.ethz.ch, yvonne-anne.pignolet@ch.abb.com

ABSTRACT

This paper presents an efficient algorithm for detecting and disseminating information in a single-hop multi-channel network: k arbitrary nodes have information they want to share with the entire network. Neither the nodes that have information nor the number k of these nodes are known initially. This communication primitive lies between the two other fundamental primitives regarding information dissemination, broadcasting (one-to-all communication) and gossiping (total information exchange). The time complexity of the information exchange algorithm we present in this paper is linear in the number of information items and thus asymptotically optimal with respect to time. The algorithm does not require collision detection and thanks to using several channels the lower bound of $\Omega(k + \log n)$ established for single-channel communication can be broken.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms, Theory

1. INTRODUCTION

For about a dozen years we have been witnessing a revolution in wireless communication, as new inexpensive near-range technology standards such as Wireless LAN or Bluetooth emerged. A lot of research has been devoted to study the problem complexity and devise algorithms for one wireless communication channel. In practice, most wireless devices can use more than one channel which allows us to solve some problems faster. We believe that this it is important to revisit basic communication primitives leveraging the availability of multiple channels. In this paper we restrict ourselves to the simplest possible network topology, the single-hop network, where every node can communicate directly with each other node, with multiple communication channels available. Imagine for example a bunch of wireless sensors monitoring an area. Sometimes, a few nodes make an observation which they need to communicate to the others. For many applications all nodes of the network should be notified of certain events efficiently, e.g., in order to raise an alarm or react to a particular situation. For such cases, we need a fast information dissemination primitive. To this end we study the problem of distributing k information items originating from k unknown sources efficiently in multi-channel networks. In other words, we generalize the *Information Exchange Problem* [9] (also known as *k-Selection* [16] and *Many-to-All Communication* [4]) for networks with several communication channels.

PROBLEM 1.1 (INFORMATION EXCHANGE).
Consider a network of n nodes with an arbitrary subset of $k \leq n$ nodes where each of these k nodes (called reporters*) is given a distinct piece of information. The* Information Exchange Problem *consists of disseminating these k information items to every node in the network. The subset of the nodes with information items is not known to the network.*

The problem complexity depends on the fact whether or not n and k is known to the participants and on the communication model. In wireless networks, messages that are transmitted on the same channel at the same time collide and cannot be decoded. Moreover, wireless devices are often not able to perform *collision detection*, i.e., they cannot distinguish ambient noise from a collision and therefore are not able to detect that a collision occurred at all. This paper explores the problem in networks without collision detection. Analogously to previous work (e.g., [16]) we study a static case where a worst-case adversary inserts k information items at the beginning of the first time slot and no more items are inserted later.

Naturally, learning new information takes time, depending on the available bandwidth and frame length. If a constant number of information items fit into one message, a lower bound on the time complexity for the Information Exchange problem is $\Omega(k)$ since at any point in time the message from at most one node can be received successfully on one channel and this message can only contain a constant number of

*Part of this research was conducted while Y.A. Pignolet was a postdoctoral fellow at the IBM Research Zurich Laboratory and at BGU Be'er-Sheva, Israel

FOMC'11, June 9, 2011, San Jose, California, USA.

information items. In this paper we propose an algorithm that solves the problem in asymptotically optimal time complexity for any (unknown) k with high probability in n.[1] In addition we construct an algorithm that solves the Information Exchange problem even if k is unknown.

One could think of a very simple algorithm to solve this problem: Estimate the number of nodes with information items k (with a size estimation algorithm, e.g. [2]) and then let all these nodes send with probability $1/k$. But even if the estimate is accurate, this approach does not guarantee the distribution of all information items to all nodes in time $O(k)$ whp_n.[2] Thus a more sophisticated method is necessary to tackle this problem efficiently and have a high success probability for all values of k.

2. OUR CONTRIBUTIONS

In a first step we assume the number of information items k to be known up to a constant, i.e., we assume that the algorithm knows a number $\tilde{k} \in \mathbb{N}$ satisfying $\tilde{k}/2 \leq k \leq 2\tilde{k}$. Later we show that this bound is not necessary.

Depending on the value of $\tilde{k}$, different strategies are applied to guarantee a timely detection and distribution of information items whp_n. More precisely, we devise two randomized algorithms, each suitable for a different range of k, and one deterministic algorithm for any k. All algorithms run in time $O(\tilde{k})$ and run correctly whp_n. The constant β depends on the desired success probability (β is independent of n and k, see Sec.8 for more details).

THEOREM 2.1 (SECTION 7). *For $\tilde{k} < \sqrt{\log n}$, Algorithm $\mathcal{A}_{\text{tiny}}$ distributes all information items in $\Theta(\tilde{k})$ time slots whp$_n$ using $O(n^{1/2})$ channels.*

THEOREM 2.2 (SECTION 8). *Let β be a constant to be chosen later (independent of n and $\tilde{k}$). For $\sqrt{\log n} \leq \tilde{k} < \frac{\log n - 3}{\beta}$, Algorithm $\mathcal{A}_{\text{small}}$ distributes all information items in $\Theta(\tilde{k})$ time slots whp$_n$ using $O(n^{\beta \log \tilde{k}/\tilde{k}})$ channels.*

THEOREM 2.3 (SECTION 6). *The deterministic Algorithm $\mathcal{A}_{\text{tree}}$ distributes all information items in time $\Theta(\max\{\tilde{k}, \log n\})$ using n channels.*

Observe that Since Algorithm $\mathcal{A}_{tree}$ is used on a small subnetwork (with accordingly smaller runtime) in Algorithm $\mathcal{A}_{small}$, we describe Algorithm $\mathcal{A}_{tree}$ in a more general form than necessary to solve the information exchange problem for $\tilde{k} > \frac{\log n - 3}{\beta}$.

Next we argue that the above algorithms can be combined to solve the selection problem for unknown k even without needing given bounds on k like $\tilde{k}/2 \leq k \leq 2\tilde{k}$. We construct Algorithm $\mathcal{A}$ using the above algorithms. We start with estimating k to be $\tilde{k} = 2$, run the appropiate algorithm for the current range of $\tilde{k}$. We repeat this process until all information items have been distributed.

[1]An event $\mathcal{E}$ occurs with high probability in x (whp_x), if $\Pr[\mathcal{E}] \geq 1 - \frac{1}{x^\alpha}$ for any fixed constant $\alpha \geq 1$. By choosing α, this probability can be made arbitrarily low. Usually one is interested in whp in n.

[2]If $k \in \Omega(\log n)$, whp_n is possible. Observe that for any $k \in o(\log n)$ this algorithm does only achieve whp_k.

THEOREM 2.4 (SECTION 9). *Algorithm $\mathcal{A}$ needs at most $\Theta(k)$ time slots after which all information items have been detected and distributed whp$_n$ even if k is unknown and no bounds on k are given.*

The number of channels our randomized algorithms need is large in order to guarantee high success probability ($\Theta(\sqrt{n})$ channels). The deterministic algorithm presented requires even more channels for a timely distribution. Such large numbers of channels are rarely available in practice. Thus we mainly view our work as a first step to generalizing the information exchange problem to multiple channels proofing that a time-optimal distribution is possible. Reducing the number of channels necessary and providing tight trade-offs between the number of channels and the time complexity is left as an open problem for future research.

The proposed algorithms can be used as a subroutine for other algorithms that disseminate information of a subset of nodes to the whole network. For example, we expect them to enable time-optimal network monitoring and cope with nodes crashing at any time for all values of k and not only for $k \in \Omega(\log n)$ as in [10].

3. RELATED WORK

The information exchange problem has been studied for single-channel networks. A non-constructive upper bound (based on the probabilistic method) was given by Komlos and Greenberg [15]. Clementi, Monti and Silvestri [6] provided a lower bound in $\Omega(k \log(n/k))$ for oblivious deterministic k-selection protocols (where the sequence of transmissions does not depend on messages received previously, this result also holds for adaptive deterministic protocols in the model without collision detection). Kowalski [16] proves the existence of an oblivious deterministic algorithm without collision detection that distributes k information items in time $O(k \log(n/k))$ based on selectors as well as a matching lower bound. Moreover, he presents an explicit polynomial-time construction with time complexity
$O(k \operatorname{polylog} n)$ to solve this problem deterministically. Later these results have been improved and extended in [4] to multi-hop networks and the authors provide bounds for centralized and distributed algorithms. In contrast to our assumptions, they assume that all k information items fit into one message and they let the nodes know how many information items are to be distributed. Furthermore, they only strive for success probabilities of at least $1 - k^{-\alpha}$ (whp_k), where we require $1 - n^{-\alpha}$ (whp_n). When restricted to single hop networks, they present a randomized algorithm that disseminates all information items in time $O(\log k(\log^2 n + k))$ whp_k. Kushilevitz and Mansour proved a lower bound of $\Omega(k + \log n)$ on the expected time of randomized algorithms [18]. The average time complexity in directed networks is addressed in [5] with bounds $O(\min(k \log(n/k), n \log n)$ and $\Omega(k/\log n + \log n)$. Furthermore, they devised a protocol for the case when information items have to be delivered separately (as in our model) within time $O(k \log(n/k) \log n)$ and a lower bound of $\Omega(k \log n)$. Exploiting the availability of multiple channels we achieve better bounds: the dissemination problem can be solved in asymptotically optimal time complexity $\Theta(k)$. Recently, Gilbert and Kowalski [9] provided upper and lower bounds for the Information Exchange

problem in single-channel networks where some of the nodes exhibit Byzantine behavior.

Apart from the Information Exchange problem many other problems are non-trivial even in single-hop networks. Other communication primitives studied for networks without collision detection are initialization (n nodes without IDs are assigned labels $1, \ldots, n$) [22], wake up [3, 8], consensus and mutual exclusion [1, 7], leader election [6, 11, 17, 19, 20, 21], size approximation [2, 11, 12], alerting (all nodes are notified if an event happens at one or more nodes) [14], sorting (n values distributed among n nodes, the i^{th} value is moved to the i^{th} node) [13], aggregation problems like finding the minimum, maximum, median value and computing the average value [23, 19].

4. COMMUNICATION MODEL

The network consists of a set of n nodes, each node v with a built-in unique ID id_v known to all other nodes (for simplicity we assume these IDs to be $\{1, \ldots, n\}$). When using an initialization algorithm that assigns IDs to nodes, e.g., [22], this assumption can be dropped. All nodes are within communication range of each other, i.e., every node can communicate with every other node directly (single-hop). To simplify the presentation of the algorithms and their analysis, we assume time to be divided into synchronized time slots. Messages are of bounded size, i.e., we assume that each message can only contain one information item. We assume that n properly divided communication channels are available (for some of our algorithms, a lower number of channels suffices). In each time slot a node v chooses a channel c and performs one of the actions `transmit` (v broadcasts on channel c) or `receive` (v monitors channel c). A transmission is successful, if exactly one node is transmitting on channel c at a time, and all nodes monitoring this channel receive the message sent. If more than one node transmits on channel c simultaneously, listening nodes can neither receive any message due to interference (called a *collision*) nor do they recognize any communication on the channel (the nodes have *no collision detection* mechanism).

5. REPORTER-FREE SET IN $O(\tilde{k})$

A building block we often use is to determine a set of nodes without reporters fast using one channel. To this end, the Procedure $\mathcal{P}_{RF}(x)$ determines a reporter-free set of size x. It starts by letting the nodes with the smallest $2\tilde{k}+2$ IDs reveal whether they are reporters by transmitting their IDs on the first channel one after another. At least two of those nodes are not reporters, because there are at most $k \leq 2\tilde{k}$ reporters altogether. The two smallest ID nodes without information to distribute are assigned to be the coordinator and the dummy node respectively (takes time $O(\tilde{k})$). A reporter-free set of size $x < (n-2-2\tilde{k})/(2\tilde{k})$ is found by letting the dummy node and the reporters out of the set of nodes with IDs $[2\tilde{k}+3, \ldots, 2\tilde{k}+3+x]$ transmit at the same time on channel 1 while the coordinator listens on channel 1. Afterwards, the coordinator informs all nodes whether it heard the dummy node, in which case a reporter-free set has been found. Otherwise, this procedure is repeated for the next set of x nodes. Since at most k sets can contain a reporter, a reporter-free set is identified within $O(k) = O(\tilde{k})$ time slots. Thus we can state the following Lemma.

LEMMA 5.1. *Procedure $\mathcal{P}_{RF}(x)$ ensures that after its completion all nodes know the IDs of a reporter-free set of size x in time $O(\tilde{k})$ using one channel if $x < (n-2-k)/k$.*

6. DETERMINISTIC DISSEMINATION ALGORITHM $\mathcal{A}_{TREE}$

We can use a balanced binary tree to disseminate information deterministically in time $O(k + \log n)$. The tree determines a schedule, where each node transmits all its messages on its own channel and children or parent nodes listen on specified channels according to the schedule. After each transmission/reception the nodes sort the messages they currently have prepare the message with the lowest reporter ID for the next transmission.

ALGORITHM 6.1.
Algorithm $\mathcal{A}_{\text{tree}}$ for each node v with ID id_v

determine position in balanced binary tree based on id_v;
while** root has not sent "stop"-message **do
    receive item from children / send next item to parent on channel id_v according to schedule S;
end while
***if** v is root **then** send information items on channel* 1*;*
***else** receive information items on channel* 1 *from root;*

The positions of the nodes are assigned as follows. Node v with $id_v = 1$ is the root and any other node v with ID id_v has a father w with ID $id_w = \lfloor id_v/2 \rfloor$. For $2 \cdot id_v \leq n$ (or $2 \cdot id_v + 1 \leq n$) the node with ID $2 \cdot id_v$ (or $2 \cdot id_v + 1$) is a child of v. The nodes exchange messages with their parents, children and the root according to a schedule consisting of five time slots which is repeated continuously until the root broadcasts a message indicating that it has received all information items. The first time slot of the schedule is assigned to the root node, all other nodes listen to the root broadcasting on channel 1. In the following four time slots each node can send one piece of information to its parent and receive one piece of information from each child: Each node v in odd levels of the tree (that is $\lfloor log_2(id_v) \rfloor$ is odd) receives one message from child $2 \cdot id_v$ in the first time slot and from child $2 \cdot id_v + 1$ in the second time slot – observe that children are in even levels. Then each node v in even levels of the tree receives one message from child $2 \cdot id_v$ in the next time slot and from child $2 \cdot id_v + 1$ in the last time slot. Every node u sends messages on its own channel u to avoid collisions – receivers will tune to this channel. The complete schedule $s_n : [n] \times \{1,2,3,4,5\} \rightarrow \{receive, send\} \times [n]$ is given by

$$s_n(id_v,1) = \begin{cases}(send,1) & : id_v = 1\\ (receive,1) & : \text{otherwise}\end{cases}$$

$$s_n(id_v,2) = \begin{cases}(receive, 2\cdot id_v) & : \lfloor \log_2(id_v)\rfloor \text{ is odd}\\ (send, id_v) & : \lfloor \log_2(id_v)\rfloor \text{ is even}\end{cases}$$

$$s_n(id_v,3) = \begin{cases}(receive, 2\cdot id_v+1) & : \lfloor \log_2(id_v)\rfloor \text{ is odd}\\ (send, id_v) & : \lfloor \log_2(id_v)\rfloor \text{ is even}\end{cases}$$

$$s_n(id_v,4) = \begin{cases}(receive, 2\cdot id_v) & : \lfloor \log_2(id_v)\rfloor \text{ is even}\\ (send, id_v) & : \lfloor \log_2(id_v)\rfloor \text{ is odd}\end{cases}$$

$$s_n(id_v,5) = \begin{cases}(receive, 2\cdot id_v+1) & : \lfloor \log_2(id_v)\rfloor \text{ is even}\\ (send, id_v) & : \lfloor \log_2(id_v)\rfloor \text{ is odd}\end{cases}$$

If a channel (vertex) on (to) which a node v should send or listen is not in the range of $\{1,\dots,n\}$, then v can be sure that the corresponding node does not exist and just sleeps in this slot – this happens if v is the root or a leaf.

The nodes use this schedule to send all information items to the root of the tree and the root can use every fifth slot to end the protocol.

We now prove Theorem 2.3 stating that $\mathcal{A}_{tree}$ distributes all information items within $O(k+\log n)$ time slots, i.e., linear in the height of a balanced binary tree.

Proof of Theorem 2.3. During one execution of the loop in lines 2–4 (lasting five time slots), each node can exchange messages with its children and parent and listen to a broadcast message of the root. To ensure that the root obtains all information items, each node v maintains a list of items. In each execution of the loop, it might receive up to two new items from its children, these items are appended to the list. Each time it sends a message to its parent using the schedule, it removes the first element of the list. The sequences of sending/receiving depends on the IDs of the nodes. Using this procedure ensures that no collisions occur as each node uses a separate channel for communication. After $height(tree)+k$ phases (in each phase the schedule above is executed) the root has received all items. The root can detect when it has received all items: If in any phase $p > height(tree)+1$ it does not receive any message, no more messages will arrive (if a child still had a message it would have sent it – and since all nodes behave like this, it follows due to the height of the tree that there are no more items stored in the lists of the other nodes by induction). When all messages have arrived, the root sends "stop" on channel 1 and transmits all information items on channel 1 subsequently. Thus each node knows all information items after $O(k+height(tree)) = O(k+\log n)$ time slots and Theorem 2.3 follows. □

7. ALGORITHM $\mathcal{A}_{TINY}$

The **basic idea** of the algorithm $\mathcal{A}_{tiny}$ is that each reporter selects a random channel from a large set of channels, such that at least half of the reporters choose a unique channel. We call a transmission of a reporter that chooses a unique channel a "successful transmission" since in this case no collision occurs. The number $K := n^{1/(2\tilde{k})}$ of channels is selected in such a way that it is small enough to ensure that for each of the $\sum_{i=0}^{\tilde{k}} \binom{K}{i}$ possible subsets of at most $\tilde{k}$ channels with a successful transmission there is a node in the network that can be assigned to listen to that subset. Each such listener then listens on all channels from the assigned subset one after another. We argue that there is a unique node (called the "boss") that listens exactly on those channels on which the information items were transmitted successfully. Thus this boss collects the information of all successful reporters (at least half of all reporters transmitted successfully) and broadcasts it subsequently. In other words, the boss can successfully transmit the gathered information to the network and thus the number of reporters is cut in half in time $O(\tilde{k})$. Repeating this procedure until no reporters are left takes time $O(\tilde{k})/2^0 + O(\tilde{k})/2^1 + O(\tilde{k})/2^2 + \cdots + O(\tilde{k})/2^{\log O(\tilde{k})} = O(\tilde{k})$ as well and thus yields Theorem 2.1. Algorithm 7.1 provides a description in pseudo-code.

Algorithm 7.1.
Algorithm $\mathcal{A}_{\text{tiny}}$ for $\tilde{k} < \sqrt{\log n}$

1: *find reporter-free set $\mathcal{L}$ of size $\sqrt{n\log n}$ with $\mathcal{P}_{RF}(\sqrt{n\log n})$;*
2: *nodes of $\mathcal{L}$ compute $\mathcal{S}^{\leq\tilde{k}} := \{S_0,\dots,S_{N-1}\}$, the set containing all N subsets of the channels $\{1,\dots,K\}$ of size at most $\tilde{k}$, where $K := 2^{\log n/(2\tilde{k})}$;*
3: *each reporter-free set node $v \in \mathcal{L}$ with ID id_v maps itself to subset $S_{id_v} \in \mathcal{S}^{\leq\tilde{k}}$;*
*//** **Send information** *\//*
4: *reporters and listeners do the following simultaneously:*
- each reporter v chooses random channel in $\{1,\dots,K\}$ and sends its information item on that channel during $\tilde{k}$ time slots.
- each node $v \in \mathcal{L}$ listens for one time slot on each channel c in its assigned subset S_{id_v}.
*//** **Identify unique boss** *\//*
5: ***if** $v \in \mathcal{L}$ received a message on all $|S_{id_v}|$ monitored channels in S_{id_v} **then***
6: *v marks itself to be a candidate;*
7: ***end if***
8: ***for** $t = \tilde{k},\dots,1$ **do***
9: *each candidate v that monitored $|S_{id_v}| = t$ channels sends its ID on channel 1, all other nodes listen on channel 1;*
10: ***end for***
*//** **Broadcast all information items** *\//*
11: ***if** ID id was broadcast in step 9 **then***
12: *node $id+1$ broadcasts "id" on channel 1;*
13: *the boss (node id) broadcasts the gathered information on channel 1 to the network;*
14: ***end if***

The first lemma bounds the collision probability of the reporters in line 5 of Algorithm 7.1.

Lemma 7.1. *If each of k reporters chooses a channel uniformly at random from $\{1,\dots,K\}$ for $K := 2^{\log n/(2\tilde{k})}$, more than $k/2$ reporters select a unique channel with probability larger than $1-n^{-1/9}$.*

PROOF. Let $p := \Pr[\text{a reporter does not choose a unique channel}]$. Although p is not independent among different reporters, it is always smaller than k/K (k nodes choose one out of K channels), no matter how many of the other reporters did not choose a unique channel. We use this property in the following analysis:

$$\begin{aligned}
&\Pr[> k/2 \text{ reporters do not choose a unique channel}] \\
&\leq \sum_{i=k/2}^{k} \binom{k}{i} \cdot p^i \cdot (1-p)^{k-i} \\
&\leq \sum_{i=k/2}^{k} 2^k \cdot (k/K)^i \cdot 1 \\
&\leq k \cdot 2^k \left(k/2^{\frac{\log n}{2\cdot\tilde{k}}}\right)^{k/2} \\
&\leq 2^{\log k} \cdot 2^k \cdot 2^{\frac{k}{2}\cdot\log k - \frac{\log n}{2\tilde{k}}\cdot\frac{k}{2}} \\
&\leq 2^{-\frac{\log n}{8} + \log k + k + \frac{k}{2}\cdot\log k} \\
&\leq n^{-1/9}
\end{aligned}$$

for large n, as $k \leq 2\tilde{k} < 2\sqrt{\log n}$. □

Using the procedure described in Section 5 we can determine a reporter-free set of size $\sqrt{n\log n}$ as $\sqrt{n\log n} < (n-2-2\tilde{k})/k$ for our range of $\tilde{k}$. There are enough nodes in $\mathcal{L}$ to assign one listener node to each element of $\mathcal{S}^{\leq\tilde{k}}$ (Line 3).

CLAIM 7.2. *All subsets $S_0, \ldots, S_{N-1}$ of $\{1, \ldots, K\}$ of size at most $\tilde{k}$ can be mapped to $\sqrt{n\log n}$ nodes for $\tilde{k} \leq \sqrt{\log n}$.*

PROOF. The total number of such subsets is $\sum_{i=0}^{\tilde{k}} \binom{2^{\frac{\log n}{2\cdot\tilde{k}}}}{i} \leq \tilde{k} \cdot \left(2^{\frac{\log n}{2\cdot\tilde{k}}}\right)^{\tilde{k}} \leq \tilde{k} 2^{\frac{1}{2}\log n} \leq \sqrt{n\log n}$ as $\tilde{k} \leq \sqrt{\log n}$. Therefore we can apply the canonical mapping using the canonical enumeration of the N subsets to a subset of all nodes in the network. □

Next, we prove that at most one listener node $v \in \mathcal{L}$ obtained all information items during the loop of Line 4 of Algorithm 7.1. This node is the boss mentioned earlier.

LEMMA 7.3. *There exists one node called* boss *that can collect the information items of all successfully transmitting reporters in time $O(\tilde{k})$.*

PROOF. Each reporter sends its information on the chosen channel $\tilde{k}$ times. Due to Claim 7.2 we can assume that a unique node $v \in \mathcal{L}$ is assigned to any subset of size at most $\tilde{k}$ of the K channels, unless v is a reporter. Let us assume that $|\mathcal{L}| = N$, i.e., no reporter node is assigned to a subset in $\mathcal{S}^{\leq\tilde{k}}$. Let i_v be the number of channels that node v is assigned to. More precisely, let node v be assigned to subset $S_{id_v} = \{c_1, \ldots, c_{i_v}\}$ of $i_v \leq k$ channels. In this case v listens to each of these i_v channels one after another for exactly one time slot. Thus there are nodes w that receive all the information of the j reporters without collisions since they listen to $S_{id_w} \supseteq J$, J being the set of these j successful reporters. Furthermore, there is a unique node that collects the information from all j successful reporters without listening to any other channels (exactly one node was assigned to this subset). □

In Lines 8–12 of Algorithm 7.1 the nodes determine the unique boss.

LEMMA 7.4. *The network can identify the unique boss with probability larger than $1 - n^{-1/9}$.*

PROOF. We call each node v that received a message on each of its i_v monitored channels a *candidate*. However, there might be several candidates. The unique boss is the unique node that listened to all successful reporters and did not listen to any other ("empty") channels. To detect the unique boss among the candidates we let each candidate v send a message on the channel specified by the number of channels i_v they monitored. More precisely, for $\tilde{k}$ time slots $t = 1, \ldots, \tilde{k}$ we ask all candidates v that monitor i_v channels to send their own ID on channel 1 at time $t = i_v$. Due to Lemma 7.1 with probability larger than $1 - n^{-1/9}$ at most half of the reporters collide and thus we can assume that the number of successful reporters $j \geq 1$. Therefore a unique boss is detected with probability larger than $1-n^{-1/9}$ because at time $t < j$ no message is received: Since $j > t \geq 1$ and therefore $j \geq 2$ there are $\binom{j}{t} \geq 2$ candidates at time $t < j$ sending on channel 1, thus if a candidate that listened to $t < j$ reporters tries to transmit a message, there is a collision with another such candidate with probability larger than $1-n^{-1/9}$. At time $t = j$ a message containing the ID of the unique boss is transmitted successfully: the unique boss sends without collision at time j. At time $t > j$ no message can be received: there is no candidate since only j reporters transmitted their information successfully. Thus no listener node v can receive a message on all $i_v = t > j$ channels. Since we have $j \leq \tilde{k}$ during the $\tilde{k}$ time slots there is exactly one time slot in which a message is sent successfully and this message contains the ID of the unique boss. Now all nodes but the boss v know, that v is the boss and the node whose ID is $id_v + 1$ informs v that it is the boss. Since we assumed that v is not a reporter it is able to broadcast all information items in Line 13. □

PROOF OF THEOREM 2.1. The probability that the boss collects the information from at least half of the reporters and can be identified is at least $(1 - n^{-1/9})$ due to Lemmas 7.3 and 7.4. Then the boss can broadcast all items it gathered from the (with probability larger than $1 - n^{1/9}$) at least $k/2$ successful reporters it is aware of on channel 1 (Line 13). Since the boss is unique no collisions occur. These transmissions take time $O(\tilde{k})$. By repeating algorithm $\mathcal{A}_{tiny}$ 9α times, we can amplify the success probability of $1-n^{-1/9}$ to exceed $1 - n^{-\alpha}$. This is whp$_n$ since we can choose the constant α arbitrarily. Thus the whole algorithm has time complexity $O(9\alpha\tilde{k}) = O(\tilde{k})$, which proves Theorem 2.1. □

8. ALGORITHM $\mathcal{A}_{SMALL}$

Basic idea: As seen in the previous section it is good to disseminate the information by first collecting all items at one specific node (the *boss*). In order to achieve this goal for the range of $\sqrt{\log n} \leq \tilde{k} < \frac{\log n - 3}{\beta}$ in $O(\tilde{k})$ time, the nodes execute four consecutive parts (the constant β is defined later). In step 1, the nodes determine which role they are going to play during the execution (there are k reporters, $n^{\beta\log\tilde{k}/\tilde{k}}$ listeners and $n - k - n^{\beta\log\tilde{k}/\tilde{k}}$ others). In step 2, each of the k reporters tries to tell a randomly picked listener

its information item (a balls-into-bins-style procedure with k balls and $n^{\beta \log \tilde{k}/\tilde{k}}$ bins). In step 3, the listeners send all collected information items to the boss. In step 4, the boss broadcasts the collected information items. Algorithm 8.1 gives an overview of the algorithm proposed in this section.
Step 1: As in $\mathcal{A}_{tiny}$ we use the procedure of Section 5 to find a set without reporters in time $O(k)$. The upper bound on $\tilde{k} < (\log n - 3)/\beta$ ensures that this procedure works for $k < 2\tilde{k}$. In the remaining three parts of the algorithm, each node executes a procedure depending on its role. The nodes that are neither reporters nor listeners wait until they are told that the information items are broadcast on channel 1 starting in the next time slot.

ALGORITHM 8.1.
Algorithm $\mathcal{A}_{\text{small}}$ *for* $\sqrt{\log n} \leq \tilde{k} < \frac{\log n - 3}{\beta}$

1: *find listener set* $\mathcal{L}$, $|\mathcal{L}| = n^{\beta \log \tilde{k}/\tilde{k}}$ *with* $\mathcal{P}_{RF}(n^{\beta \log \tilde{k}/\tilde{k}})$;
2: ***for*** $i := 1, \ldots, \tilde{k}$ ***do***
  if *reporter* ***then*** *transmit information item on random channel among* $\{1, \ldots, |\mathcal{L}|\}$;
  else if *listener* ***then*** *listen on assigned channel and create set of information items received;*
3: ***if*** *listener* ***then*** *forward collected items to boss with tree dissemination algorithm* $\mathcal{A}_{\text{tree}}$;
4: ***if*** *boss* ***then*** *broadcast all information items on channel 1;*
  else *listen on channel 1;*

Step 2: The reporters try to transmit their information items to the listeners by a randomized "balls into bins"-style procedure repeated $\tilde{k}$ times. Each of the $\tilde{k}$ reporters chooses a channel c uniformly at random from $[1, n^{\beta \log \tilde{k}/\tilde{k}}]$ to send its information item, while each of the $n^{\beta \log \tilde{k}/\tilde{k}}$ listener nodes listens on a unique channel (throwing a ball at random into a bin). A "listening time slot" is called successful if a listener l_I has received an item U_J.

In each of the $\tilde{k}$ trials, a reporter is successful whp$_n$, thanks to the bound $k < 2\tilde{k}$ and $\tilde{k} \geq \sqrt{\log n}$.

CLAIM 8.1. *Since* $k < 2\tilde{k}$ *the probability that a fixed reporter* v *is able to transmit* U_I *to a listener during the* $\tilde{k}$ *repetitions of step 2 is at least* $1 - 1/n^{\beta-1}$.

PROOF. At first, we want to bound Pr[v is not successful in the first round] for a reporter v. Again, this probability is not independent among different reporters. But Pr[v is not successful in the first round] is at most $2\tilde{k}/n^{\beta \log \tilde{k}/\tilde{k}} < 1 - n^{(\beta-1) \log \tilde{k}/\tilde{k}}$ since each of the $|\mathcal{R}| \leq 2\tilde{k}$ reporters chooses one of $|\mathcal{L}| = n^{\beta \log \tilde{k}/\tilde{k}}$ channels uniformly at random, no matter how many of the other reporters choose the same channel. Hence we derive that Pr[v is not successful in all $\tilde{k}$ rounds] is less or equal then $\left(1/n^{(\beta-1) \log \tilde{k}/\tilde{k}}\right)^{\tilde{k}} \leq 1/n^{\beta-1}$. □

Reporters do not need to be notified if their transmission was successful as all items are transmitted whp$_n$. The reporters keep sending their items even if they have already been detected by a listener.

LEMMA 8.2. *If* $k < 2\tilde{k}$ *then the probability that all reporters successfully transmitted their information to the listener nodes* $\mathcal{L}$ *is at least* $1 - n^{-(\beta-2)}$.

PROOF. The probability that all reporters transmitted their information successfully after $\tilde{k}$ rounds is equal to Pr[After $\tilde{k}$ rounds each reporter v successfully transmitted U_I to a listener] which can be lower bounded by $\left(1 - 1/n^{\beta-1}\right)^{\tilde{k}}$ applying claim 8.1 and the initial assumption that there are $k < 2\tilde{k}$ reporters. Hence, the above-mentioned probability is at least $1 - \frac{\tilde{k}}{n^{(\beta-1)}}$ due to Bernoulli's inequality (stating that $(1+x)^y \geq 1 + yx$ for every integer $y > 0$ and every real number $x \geq -1$). This probability is at least $1 - n^{-(\beta-2)}$ since $\tilde{k} \leq 2k \leq n$, and the lemma follows. □

As long as each reporter can transmit to a listener successfully during the $\tilde{k}$ repetitions of the balls-into-bins procedure, the algorithm works correctly. If one or more reporters are not known to the boss after this procedure, the algorithm fails. This failure probability p is, as we just proved in Lemma 8.2, upper bounded by $n^{-(\beta-2)}$.
Step 3: Inform the boss The reporters sleep while the listeners forward the collected information items to their boss using the tree dissemination algorithm $\mathcal{A}_{tree}$ presented in Section 6. This takes time $O(\tilde{k})$ since the time to disseminate the k information items via the tree dissemination algorithm in a network of size $|\mathcal{L}| = n^{\beta \log \tilde{k}/\tilde{k}}$ is in $O(\log |\mathcal{L}| + k) = O(\log(n \cdot \beta \log \tilde{k}/\tilde{k}) + \tilde{k}) = O(\tilde{k})$ for all $\tilde{k} \in \Omega(\sqrt{\log n})$.

LEMMA 8.3. *For* $\tilde{k} \in \Omega(\sqrt{\log n})$ *all reporters are known to the boss whp*$_n$ *after* $O(\tilde{k})$ *time slots.*

PROOF. This follows from the fact that the listeners can disseminate all information items to the boss in $O(\tilde{k})$ time slots as we just showed and from Lemma 8.2: Let the desired success probability of Algorithm $\mathcal{A}_{small}$ be $1 - n^{-\alpha}$. If α is a constant which can be chosen arbitrarily to make this probability arbitrarily large, this is whp$_n$. Now, if we choose $\beta = \alpha + 2$, we obtain that for $\tilde{k} \in \Omega(\sqrt{\log n})$, all reporters can report to a listener with probability at least $1 - n^{-\alpha}$. □

Step 4: Broadcast information items The listener node specified to be the boss of $\mathcal{L}$ has collected all information items and broadcasts the information items it obtained on channel 1. No collisions occur and the time complexity of this step is $O(\tilde{k})$.

We are now ready to prove Theorem 2.2.

PROOF OF THEOREM 2.2. In step 1, each node needs $O(\tilde{k})$ time to decide whether it is a listener, reporter or other. In step 2, the Algorithm 8.1 performs $\tilde{k}$ repetitions of the "balls into bins" procedure—each repetition takes 2 time slots. Since $k < 2\tilde{k}$ then the boss receives the information items of all nodes whp$_n$ thanks to Lemma 8.3 in step 3. Finally, in step 4, all the collected information items are exchanged, which requires $O(\tilde{k})$ time slots as well. The number of channels required is bounded by the number of listeners $O(n^{\beta \log \tilde{k}/\tilde{k}})$. □

9. ALGORITHM $\mathcal{A}$ FOR UNKNOWN k

Until now we considered algorithms that need a lower and upper bound on the actual number of information items k:

$\tilde{k}/2 \leq k \leq 2\tilde{k}$. In this section we present an algorithm $\mathcal{A}$ that works for arbitrary values for k without any bounds on k given in advance. To this end it uses an estimate $\tilde{k}$ of k that is set to $\tilde{k} = 2$ in the beginning and doubled until reaching k. Note that the algorithms $\mathcal{A}_{tiny}$ and $\mathcal{A}_{small}$ are still able to finish, but depending on the size of k compared to $\tilde{k}$ none or not all messages might get through. Yet all nodes have obtained the same information afterwards.

For each value $\tilde{k}$ Algorithm $\mathcal{A}$ uses the appropriate algorithm $\mathcal{A}_{tiny}$, $\mathcal{A}_{small}$ or $\mathcal{A}_{tree}$ as a subroutine. After the completion of this subroutine, the dummy node 2 and the reporters that have not been able to distribute their message transmit simultaneously on channel one. In the subsequent time slot the boss notifies the network that $\tilde{k}$ was too small (Line 6), every participant doubles $\tilde{k}$ and the procedure is repeated for the remaining reporters.

ALGORITHM 9.1.
Algorithm $\mathcal{A}$ for Unknown k

each node:

1: ***if*** *node 1 has no information item* ***then*** *inject dummy-information at node 1;*
2: $\tilde{k} := 2$; ***//** estimate for*** k
3: $tooSmall := true$;
4: ***while*** $\tilde{k} \leq \frac{\log n - 3}{\beta}$ ***and*** $tooSmall$ ***do***
5: ***if*** $\tilde{k} < \sqrt{\log n}$ ***then*** $\mathcal{A}_{tiny}()$;
else $\mathcal{A}_{snall}()$;
6: $tooSmall := false$ *if all reporters successful;*
7: $\tilde{k} := 2\tilde{k}$; ***//** double estimate***
8: ***end while***
9: ***if*** $tooSmall$ ***then*** $\mathcal{A}_{tree}()$;

Since the time complexity of the algorithms using the estimate $\tilde{k}$ is linear in $\tilde{k}$ (since it is used only in the indicated range of k – Line 5) and can detect whp_n whether $\tilde{k}/2 \leq k \leq 2\tilde{k}$ or not, the runtime of the final algorithm is $O(1 + 2^1 + 2^2 + \cdots + 2^{\log k - 1} + k) = O(k)$ whp_n.

In order to distinguish between the case without any information items and cases with at least one item, we insert a "dummy" item at node 1 (if node 1 does not have an item to disseminate already, see Line 1 of Algorithm 9.1). We assume that the "dummy" item can not be injected by an adversary (for example by using special symbols the adversary is not allowed to use). Thanks to this trick we artificially ensure that $k \neq 0$. This enables us to overcome the problem that Algorithm 7.1 $\mathcal{A}_{tiny}$ cannot distinguish the case $k = 0$ where no information has to be spread from the case $2\tilde{k} < k$. Thus the "dummy item" prevents Algorithm 9.1 $\mathcal{A}$ from doubling the estimate if $k = 1$ because Algorithm 7.1 $\mathcal{A}_{tiny}$ can detect that there are no messages if the dummy-message is the only message that was disseminated. This has no impact on the time complexity, but ensures that Algorithm $\mathcal{A}$ can detect that there are no items to disseminate in time $O(1)$ whp_n. Theorem 2.4 follows from these observations.

Our Information Exchange Algorithm can also be extended for the dynamic setting as proposed in [16], where nodes might obtain new information items during the execution of the algorithm. To this end, the eight steps from Algorithm 9.1 are repeated in an endless loop. Node 1 broadcasts a "start"-message at the beginning of each such loop, and nodes which get a new item U within one loop ignore it until the next broadcast of a start message. Then, they start trying to disseminate their new item to the other nodes. Using this method the algorithm is able to distribute all information items with a latency of $\Theta(k)$.

10. CONCLUSION

In this paper, we considered the problem of disseminating information in a single-hop multi-channel network after k nodes have received an information item to be distributed among all n nodes in the network. We described different algorithms which perform well for different numbers of such information items without needing the ability to detect collisions. These algorithms can be combined such that we obtain an algorithm that is guaranteed to disseminate all information items to all nodes within $\Theta(k)$ time with high probability in n, which asymptotically optimal if messages cannot be merged. If we assume that the energy consumption of transmitting and receiving is in the same order of magnitude, the protocol is also asymptotically optimal with respect to energy. In the way we described our algorithm, a few nodes (for example the boss of the listeners) have to be awake during more time slots than most of the other nodes. However, it is easy to achieve a balanced energy consumption among all nodes by using simple tricks, such as the nodes taking turns in being the boss. The algorithm can be used as a subroutine for other algorithms that disseminate information of a subset of nodes to the whole network. For example, we expect it to enable time-optimal network monitoring and cope with nodes crashing at any time during the execution of the algorithm for all values of k and not only for $k \in \Omega(\log n)$ as in [10].

11. REFERENCES

[1] Marcin Bienkowski, Marek Klonowski, Miroslaw Korzeniowski, and Dariusz R. Kowalski. Dynamic Sharing of a Multiple Access Channel. In *27th Int. Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 83–940, 2010.

[2] I. Caragiannis, C. Galdi, and C. Kaklamanis. Basic computations in wireless networks. In *ISAAC 05*, volume 3827, page 533–542, 2005.

[3] B. Chlebus and L. Gasieniec. On the Wake-Up Problem in Radio Networks. In *Automata, languages and programming: 32nd international colloquium, ICALP*, page 347, 2005.

[4] B. Chlebus, D. Kowalski, and T. Radzik. Many-to-Many Communication in Radio Networks. In *Algorithmica*, volume 54:1, pages 118–139, 2009.

[5] B. Chlebus, D. Kowalski, and M. Rokicki. Average-time complexity of gossiping in radio networks. In *Structural Information and Communication Complexity*, pages 253–267, 2006.

[6] A. Clementi, A. Monti, and R. Silvestri. Distributed broadcast in radio networks of unknown topology. In *Theoretical Computer Science*, 302(1-3):337–364, 2003.

[7] J. Czyzowicz, L. Gasieniec, D. R. Kowalski, and A. Pelc. Consensus and mutual exclusion in a multiple

access channel. In *Distributed Computing*, pages 512–526, 2009.

[8] L. Gasieniec, A. Pelc, and D. Peleg. The wakeup problem in synchronous broadcast systems (extended abstract). In *Proceedings of the ACM symposium on Principles of distributed computing (PODC)*, pages 113–121, 2000.

[9] S. Gilbert and D. Kowalski. Trusted Computing for Fault-Prone Wireless Networks. In *Distributed Computing*, pages 359–373, 2010.

[10] S. Holzer, Y. Pignolet, J. Smula, and R. Wattenhofer. Monitoring Churn in Wireless Networks. In *Algorithms for Sensor Systems*, pages 118–133, 2010.

[11] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Energy-efficient size approximation of radio networks with no collision detection. In *8th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 279–289, 2002.

[12] J. Kabarowski, M. Kutylowski, and W. Rutkowski. Adversary immune size approximation of single-hop radio networks. In *Theory and Applications of Models of Computation (TAMC)*, 3959:148–158, 2006.

[13] M. Kik. Merging and Merge-Sort in a Single Hop Radio Network. In *Theory and Practice of Computer Science (SOFSEM)*, pages 341–349, 2006.

[14] M. Klonowski, M. Kutyłowski, and J. Zatopianski. Energy Efficient Alert in Single-Hop Networks of Extremely Weak Devices. In *Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 139–150, 2009.

[15] J. Komlos and A. Greenberg. An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. In *Information Theory, IEEE Transactions on*, 31(2):302–306, 1985.

[16] D. Kowalski. On selection problem in radio networks. In *Proceedings of the ACM symposium on Principles of distributed computing (PODC)*, pages 158–166, 2005.

[17] D. Kowalski and A. Pelc. Leader Election in Ad Hoc Radio Networks: A Keen Ear Helps. In *International Conference on Automata, Languages and Programming (ICALP)*, page 521–533, 2009.

[18] E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ Lower Bound for Broadcast in Radio Networks. In *SIAM Journal on Computing*, 27:702, 1998.

[19] M. Kutyłowski and D. Letkiewicz. Computing Average Value in Ad Hoc Networks. In *Mathematical Foundations of Computer Science (MFCS)*, 2747:511–520, 2003.

[20] M. Kutyłowski and W. Rutkowski. Adversary Immune Leader Election in Ad Hoc Radio Networks? In *European Symposium on Algorithms (ESA)*, pages 397–408, 2003.

[21] C. Lavault, J. Marckert, and V. Ravelomanana. Quasi-optimal energy-efficient leader election algorithms in radio networks. In *Information and Computation*, 205(5):679–693, 2007.

[22] K. Nakano and S. Olariu. Energy-efficient initialization protocols for radio networks with no collision detection. In *IEEE Transactions on Parallel and Distributed Systems*, 11(851-863):4, 2000.

[23] M. Singh and V. K. Prasanna. Energy-optimal and energy-balanced sorting in a single-hop wireless sensor network. In *IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pages 50–59, 2003.