Conflict Resolution and Membership Problem in Beeping Model

Bojun Huang
bojhuang@microsoft.com
Microsoft Research, Beijing, China

Thomas Moscibroda
moscitho@microsoft.com
Microsoft Research, Beijing, China

Abstract

Consider a group of nodes connected through multiple-access channels and the only observable feedback on the channel is a binary value: either one or more nodes have transmitted (busy), or no node has transmitted (idle). The channel model thus described is called *Beeping Model* and captures computation in hardware using a group of sequential circuit modules connected by a logic-OR gate. It has also been used to study chemical signaling mechanisms between biological cells and carrier-sensing based wireless communication.

In this paper, we study the distributed complexity of two fundamental problems in the Beeping Model. In both problems, there is a set of nodes each with a unique identifier $i \in \{1, 2, \ldots, n\}$. A subset of the nodes $A \subseteq \{1, 2, \ldots, n\}$ is called *active nodes*. In the *Membership Problem*, every node needs to find out the identifiers of all active nodes. In the *Conflict Resolution Problem*, the goal is to let every active node use the channel alone (without collision) at least once.

We derive two results that characterize the distributed complexity of these problems. First, we prove that in the Beeping Model the two above problems are equally hard. This is in stark contrast to traditional channel models with ternary feedback in which the membership problem is strictly harder than conflict resolution. The equivalence result also leads to a randomized lower bound for conflict resolution, which shows a relative powerlessness of randomization in the beeping model. On the other hand, we observe that *parallelization* could be particularly powerful in Beeping Model, and our second result is to propose a novel deterministic parallel algorithm that takes significantly less time than previous solutions using a reasonable number of parallel channels.

1 Introduction

Consider an *OR* logic-gate in hardware circuit with n inputs, each controlled by a circuit module that also listens to the output of the OR-gate, as illustrated by Figure 1. Assume that all the circuit modules are synchronized by a global clock. If within a time slot *at least one* module inputs signal “1” to the OR gate, all the n modules will receive an output “1” from the gate at the end of the time slot. The simple circuit structure thus described is a physical embodiment of the distributed computing model called *Beeping Model* [6]. In this model, computing entities or nodes communicate using a shared communication channel. Instead of directly sending messages, a node can only congest the communication channel with a “beep” signal. The beep signal is then broadcast to all nodes connected to the channel. The listening nodes do not distinguish different beep signals or demodulate the mixed signals from simultaneous beeping, but only receive a *binary feedback* from the channel: “busy” if at least one node is beeping; and “idle” otherwise.

The Beeping Model has recently found a lot of attention in the distributed computing community. From several independent studies, researchers found that the Beeping Model may encompass a surprisingly wide range of primitive communication scenarios. For instances, the model was originally proposed to formulate

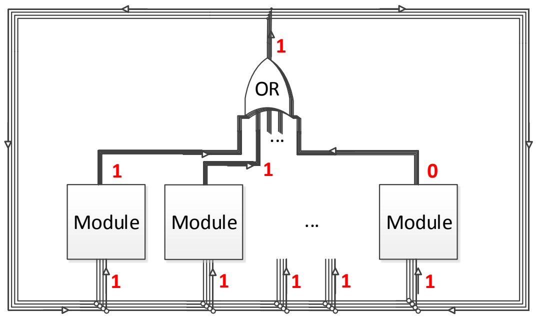

Figure 1: The hardware circuit correspondence of the Beeping Model

the carrier-sensing based radio communications for wireless network initializations [6]. Almost at the same time, a variant of the Beeping Model was also used to formulate a protein-based signaling mechanism between biological cells [2]. In our work, the practical motivation has been to build high-speed custom circuits, [1] and it turns out that the OR-gate based architecture shown in Figure 1 occurs as a core component in our design, as well as in many *circuit-based algorithm* implementations.

On the other hand, the Beeping Model is also one of the fundamental models for *multiple access channels*. Such models can be distinguished by the feedback the participants (nodes) can sense from the channel. In the traditional channel model (also called *model with collision detection*) [19] [10] [4] [17] [9] [20] [13] [14], nodes receive ternary feedback about the channel state: no one is sending (*idle*), exactly one node is sending (*success*), or two or more nodes are sending (*collision*). In contrast, the Beeping Model has only binary feedback: no one is sending (*idle*) or, one or more nodes are sending (*busy*). In Beeping Model nodes (such as cells or transistor blocks) are not assumed to be capable of distinguishing exclusive and collided signals, a capability that usually depends on complex modulation technologies.

Despite of its primitiveness, it has been shown that the Beeping Model could still allow efficient distributed computing sometimes. For example, several distributed graph problems (e.g., MIS [1], coloring [6]) have been studied in the Beeping Model, and simple but efficient beeping algorithms were proposed to solve the interval-coloring problem [6] and maximal independent set (MIS) problem [1] [2] in networks with unknown topology. Notably, all these previous works have focused on *Monte Carlo* randomized algorithms, which promise to solve problem subject to an (exponentially) small failure probability.

Membership Problem and Conflict Resolution: In this paper, we study the distributed complexity of two fundamental problems in the Beeping Model. Let $N = \{1, 2, \dots, n\}$. In both problems, there is a group of nodes each with a unique identifier (ID) $i \in N$. A subset of the nodes $A \subseteq N$ of size $|A| \leq k$ is called *active nodes*. Only active nodes can send out beep signals. Initially, each node knows whether itself is active, as well as its ID i, the size of the name space n, and the upper bound k. [2]

- The *Conflict Resolution Problem* (CR), also called k*-Selection Problem*, asks to coordinate the nodes' accesses to the channel such that for any $A \subseteq N$, each active node $i \in A$ obtains *exclusive access* to the

[1]More specifically, our goal is to build custom machine to play the game of GO, one of the few classic board games in which computers are still inferior to the best human players [8].

[2]In case either i or n is unknown, we can apply standard tricks of letting node choose random IDs (when i is unknown) or exponentially estimating n based on the IDs in the system (when n is unknown). If such a k is unknown, we just iteratively run an algorithm with $k = 2^r$ in round r, until k is large enough to be an upper bound.

channel in at least one time slot if $|A| \leq k$. A node $i \in N$ obtains exclusive access to the channel in time slot t if and only if i is the only node *allowed to beep* in time slot t. A time slot t is *exclusively-used* if and only if any $i \in N$ obtains exclusive access to the channel in time slot t. In the Beeping Model, the CR problem further asks every node $i \in N$ to correctly recognize all such exclusively-used time slots (since otherwise they cannot distinguish the real message from noise).

- The *Membership Problem* (MP), also called *Node Identification* or *Station Identification*, asks to learn the set A if $|A| \leq k$, i.e., to let every node $i \in N$ know the IDs of all the (at most k) active nodes.

We derive two results that characterize the distributed complexity of these problems. First, we prove that in the Beeping Model the two problems are equally hard for both deterministic and randomized algorithms, which is unlike in traditional access channel models in which the membership problem is strictly harder than conflict resolution. Moreover, we show that in Beeping Model *randomization* may not be as powerful as in other channel models. Specifically, we prove a randomized lower bound that matches the known deterministic upper bound, for both problems. On the other hand, we observe that *parallelization* could be particularly powerful in Beeping Model, and our second result is to propose a novel *deterministic* parallel algorithm that takes significantly less time than previous solutions using a reasonable number of parallel channels. We now discuss these contributions in detail.

Equivalence of MP and CR: Intuitively, one may think that the Membership Problem requires more time than Conflict Resolution for at least two reasons: First, in CR, an active node that has managed to successfully transmit can stop sending out any additional beeps; while this is not the case in MP. In other words, while the CR problem merely asks to arrange k "successful" time slots, the MP further asks to identify *who* is beeping in each successful time slot. Secondly, reducing CR to MP is trivial (by simply letting the active nodes beep successfully one-by-one in the order of their IDs), but the reversed reduction from MP to CR comes at an extra cost: A node can only transmit a single beep during one iteration of a CR protocol, which is not sufficient to transmit an entire identifier of length $O(\log n)$ bits, as required in MP.

Interestingly, the intuition that MP is strictly harder than CR is known to be true in the traditional ternary-feedback (idle, success, collision) model of access channels. Specifically, there is a known separation of the two problems for randomized algorithms. With ternary feedback, the expected running time of any Las Vegas membership algorithm is $\Omega(k \log \frac{n}{k})$ (by the entropy argument), while there are Las Vegas collision resolution algorithms with expected running time of $\leq 2.89k$ [19] [17].

In this paper, we show that the above intuition does not hold in the beeping model, i.e., that the membership problem is *not* harder than conflict resolution. This reveals two fundamental differences between the two basic models for multiple access channels: the binary Beeping Model and the ternary traditional model. (i) First, whereas MP is strictly harder than CR in the ternary model, the two problems are equally hard in the binary Beeping Model, which means the only way to achieve reliable (or collision-free) communication in the Beeping Model is to identify all the nodes competing for the channel. (ii) And secondly, we prove that there is a difference between the two models in the power of randomization for reliable communication. Specifically, Greenberg and Winograd proved in [9] a lower bound of $\Omega(k \log_k n)$ for any deterministic conflict resolution algorithm in the ternary model, which established the separation between deterministic algorithms and randomized algorithms in this problem (recall that there are $\Theta(k)$ randomized algorithms in the ternary model [19] [17]). In this paper we show that this gap disappears in the Beeping Model. We prove that in the Beeping Model any deterministic conflict resolution algorithm is also an algorithm that solves MP. This result yields the lower bound $\Omega(k \log \frac{n}{k})$ for CR with respect to both deterministic and randomized algorithms in the beeping model, which is tight since there exist deterministic algorithms that run in $O(k \log \frac{n}{k})$ time [7]. Table 1 summarizes the performance bounds in the two models. Results marked with asterisks are new. Finally, our proof techniques are nontrivial. As a by-product, we prove that in the

Table 1: Known Bounds on the (Sequential) Time Complexity for CR and MP

	Ternary Feedback – $\{0, 1, 2^+\}$	Beeping Model – $\{0, 1^+\}$
Membership Problem (rand.)	$\Theta(k \log \frac{n}{k})$	$\Theta(k \log \frac{n}{k})$
Collision Resolution (rand.)	$\Theta(k)$	$\Theta(k \log \frac{n}{k})$ **(*)**
Membership Problem (det.)	$\Theta(k \log \frac{n}{k})$	$\Theta(k \log \frac{n}{k})$
Collision Resolution (det.)	$\Omega(k \log_k n)$, $O(k \log \frac{n}{k})$	$\Theta(k \log \frac{n}{k})$ **(*)**

Beeping Model, one cannot count the number of active nodes without identifying them (i.e. solving MP).

Efficient Parallel Algorithm: So far, we have assumed that there is a single channel connecting the nodes. In many applications, however, nodes can access more than one beeping channel in parallel. For example, hardware circuits are typically 32-bit or 64-bit wide (i.e., there are 32 or 64 beeping channels in parallel, see Figure 1); chemical interactions between biological cells may be activated by multiple types of proteins; and a wireless communication channel may be partitioned into multiple sub-bands (e.g., OFDM used in Wi-Fi partitions each channel into so-called *sub-carriers* that are all accessed simultaneously). In each time-slot, a node can decide to beep or not independently in *each* of the beeping channels, and listen to the feedbacks of *all* channels at the same time.

Clearly, the number of channels plays an important role in how much time is required to solve the problems. For example, with n parallel channels, both problems can be solved in $O(1)$ time with a simple round robin algorithm. However, this is unrealistic as in many applications n is the size of the name space, which may grow exponentially with the length of node identifiers (e.g., $n = 2^{64}$ for 64-bit identifiers). For this reason, we seek efficient parallel algorithms that use $\text{polylog}(n)$ number of channels and have $\text{polylog}(n)$ computational complexity (e.g. avoiding full scans of the whole name space). Seems the fastest such solutions in the literature is by Chou Hsiung Li [15], in which an efficient algorithm was proposed in the context of experimental variables screening. Li's algorithm turns out to be essentially a parallel algorithm which, when used in beeping model, terminates in $O(\log \frac{n}{k})$ time with $O(k)$ channels, and has computational complexity of $O(k^2 \log \frac{n}{k})$.

The second main contribution of this paper is a novel and practical *deterministic* algorithm for both CR and MP. The basic idea of the algorithm is to iteratively reduce the problem size n by renaming each node to a smaller name space in each iteration. We show that when the algorithm terminates, each active node has an ID $i' \in \{1, \ldots, k\}$, and that each of them can locally recover the original IDs of all active nodes from the channel feedback history. The algorithm terminates in $O(\log k \log \log_k n)$ time slots in the worst case, which is *exponentially* better than Li's algorithm [15] for $k \in \text{polylog}(n)$. The algorithm uses $O(k \log_k n + k^2)$ parallel channels, and the computational complexity of the algorithm is $O(k^2 \log \frac{n}{k} + k^3)$ – both are logarithmic in n and polynomial in k. Table 2 summarizes our results relative to previous work.

In addition to its efficiency, our algorithm is also tolerant to arbitrary crash-failures in the parallel model, and always correctly returns the set of nodes that remain active when the algorithm terminates. Finally, the core component of the algorithm is a *strong renaming/coloring* process, which may be of interest in its own right. The strong renaming problem asks to assign each active node a unique ID $i' \in \{1, ..., d\}$, where d is the number of active nodes. Our algorithm, when used as a strong renaming algorithm, is *invertible* and *order-preserving* (i.e. for any two original ID's $i < j$, we have $i' < j'$).

Notations: Let $\gamma \in \{0, 1\}^*$ be a bit vector, we denote $|\gamma|$ as the length of γ, $||\gamma||$ as the number of bit "1" in γ, and $\gamma[i]$ for $i \in \{1, \ldots, |\gamma|\}$ as the i-th bit of γ. For a set of bit vectors $\gamma_1, \gamma_2, ..., \gamma_n$, $(\gamma_1, \cdots, \gamma_n)$ is the concatenation of these n vectors; $\gamma_i \vee \gamma_j$ is the bitwise Boolean Sum (i.e. logical-OR) of γ_i and γ_j; ϵ denotes the empty vector. For a natural number n, $[n]_q$ denotes the q-nary representation of n.

Table 2: Algorithms for CR and MP in Beeping Model with Multiple Channels

	# Time Slots	# Channels	Computation
Round-robin	$O(1)$	$O(n)$	$O(n)$
Adaptive GT [10] [4] [20] [7]	$O(k \log \frac{n}{k})$	$O(1)$	$O(k \log \frac{n}{k})$
Li's Algorithm [15]	$O(\log \frac{n}{k})$	$O(k)$	$O(k^2 \log \frac{n}{k})$
Our Algorithm (*)	$O(\log k \log \log_k n)$	$O(k \log_k n + k^2)$	$O(k^2 \log \frac{n}{k} + k^3)$

2 The Equivalence of Membership and Conflict Resolution

In this section we show the equivalence between MP and CR in the Beeping Model. The equivalence leads to a tight lower bound for both problems, and for both deterministic and randomized algorithms (Las Vegas and Monte Carlo). As discussed, both the equivalence of the problems and the relative powerlessness of randomization are in contrast to the traditional ternary channel access model. Without loss of generality, we assume the model has single channel in this section. We denote a problem instance (of any problem considered here) by a bit vector $\pi \in \{0, 1\}^n$, where $\pi[i] = 1$ means node i is active, and $\pi[i] = 0$ otherwise. For any deterministic algorithm $\mathcal{A}$, we denote by the bit vector $r_{\mathcal{A}}(\pi)$ the channel feedback history of algorithm $\mathcal{A}$ under problem instance π, where $r_{\mathcal{A}}(\pi)[t] = 1$ means the node hear a beep signal ("busy") from the channel in time-slot t.

The reduction from conflict resolution to membership is straightforward – once every node knows who is active, active nodes can send messages one by one without any conflict in k time-slots. However, an efficient reduction in the opposite way is non-trivial because through a single successful transmission a node can only convey 1 bit of information. A conflict resolution algorithm enables each node to transmit once, which seems insufficient to communicate a full $O(\log n)$-bit node ID, as required in the membership problem. We nevertheless show that the two problems are equivalent by resorting to reduce the membership problem to an intermediary problem, the *counting problem*, in which each node has to learn the exact *number of active nodes*. We prove that a conflict resolution algorithm can be used to solve the counting problem, and that–in the beeping model–every counting algorithm effectively solves the membership problem. This implies that instead of letting every active node explicitly report its $O(\log n)$-bit ID, in the beeping model we can infer every active node's ID as long as each of them can transmit *one single bit* successfully.

The following arguments are based on a general property of the beeping model, presented by Lemma 2.1. It asserts that, if any deterministic algorithm $\mathcal{A}$ generates the same channel feedback for two instances π and π', it must also generate exactly the same channel feedback for the instance $\pi \vee \pi'$. In other words, the equivalent class of π with respect to $r_{\mathcal{A}}(\pi)$ must be a *closure* under the logical-OR operation. The key insight behind Lemma 2.1 is that active nodes act according to the channel feedback history. Given the same feedback history before time-slot t, each active node in $\pi \vee \pi'$ is also either active in π or in π' (or in both), so none of them can lead to a different channel feedback at time t.

Lemma 2.1. *In the beeping model, for any deterministic algorithm $\mathcal{A}$, let π and π' be two instances of the problem to be solved, if $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi')$, then $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi \vee \pi')$.*

Proof. The proof is by induction. Given algorithm $\mathcal{A}$, let $\gamma_t(\pi) =< r_{\mathcal{A}}(\pi)[1], r_{\mathcal{A}}(\pi)[2], ..., r_{\mathcal{A}}(\pi)[t] >$ be the channel feedback history of $\mathcal{A}$ under π until time-slot t. So $\gamma_t(\pi)$ is a prefix of $r_{\mathcal{A}}(\pi)$ if $\mathcal{A}$ is still running in time-slot t and $\gamma_t(\pi) = r_{\mathcal{A}}(\pi)$ if $\mathcal{A}$ has terminated before time-slot t. To prove $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi \vee \pi')$, it is sufficient to prove $\gamma_t(\pi) = \gamma_t(\pi \vee \pi')$ for any $t \geq 1$.

If node i is inactive, it keeps quiet all the time; if node i is active, its decision to beep or not at time $t + 1$ fully depends on γ_t. By the indicator function $G_i(\gamma_t)$ we denote the decision of node i at time $t + 1$, where

$G_i(\gamma_t) = 1$ if node i chooses to beep and $G_i(\gamma_t) = 0$ if it keeps quiet. Note that G_i is determined once the deterministic algorithm $\mathcal{A}$ is given. By the definition of the Beeping Model we have

$$\gamma_{t+1}(\pi) = \Big(\gamma_t(\pi), \textstyle\bigvee_i \pi[i] \cdot G_i(\gamma_t(\pi))\Big), \tag{1}$$

which also holds for π' and $\pi \vee \pi'$.

Clearly we have $\gamma_1(\pi) = \gamma_1(\pi \vee \pi') = \epsilon$, since there is no feedback history at the first time slot. By induction, suppose at time t we have $\gamma_t(\pi) = \gamma_t(\pi \vee \pi')$, we only need to prove $\gamma_{t+1}(\pi) = \gamma_{t+1}(\pi \vee \pi')$, or equivalently, by Eq.(1), to prove

$$\textstyle\bigvee_i \pi[i] \cdot G_i(\gamma_t(\pi)) = \bigvee_i (\pi \vee \pi')[i] \cdot G_i(\gamma_t(\pi \vee \pi')). \tag{2}$$

Since $r_A(\pi) = r_A(\pi')$, we have $\gamma_{t+1}(\pi) = \gamma_{t+1}(\pi')$ for any $t \geq 1$, which means, again by Eq.(1),

$$\textstyle\bigvee_i \pi[i] \cdot G_i(\gamma_t(\pi)) = \bigvee_i \pi'[i] \cdot G_i(\gamma_t(\pi')). \tag{3}$$

Combining Eq.(3) and the condition that $\gamma_t(\pi) = \gamma_t(\pi') = \gamma_t(\pi \vee \pi')$, we arrive at Eq. (2) after the following transformations:

$$\begin{aligned}
\textstyle\bigvee_i \pi[i] \cdot G_i(\gamma_t(\pi)) &= \Big(\textstyle\bigvee_i \pi[i] \cdot G_i(\gamma_t(\pi))\Big) \vee \Big(\textstyle\bigvee_i \pi'[i] \cdot G_i(\gamma_t(\pi'))\Big) \\
&= \textstyle\bigvee_i \Big(\pi[i] \cdot G_i(\gamma_t(\pi))\Big) \vee \Big(\pi'[i] \cdot G_i(\gamma_t(\pi'))\Big) \\
&= \textstyle\bigvee_i \Big(\pi[i] \cdot G_i(\gamma_t(\pi))\Big) \vee \Big(\pi'[i] \cdot G_i(\gamma_t(\pi))\Big) \\
&= \textstyle\bigvee_i (\pi[i] \vee \pi'[i]) \cdot G_i(\gamma_t(\pi)) \\
&= \textstyle\bigvee_i (\pi \vee \pi')[i] \cdot G_i(\gamma_t(\pi \vee \pi')).
\end{aligned}$$

□

Some problems can be defined by a function of π, denoted by $\lambda(\pi)$ here, so that the goal of the problem is to let every node learn the value of $\lambda(\pi)$. For example, $\lambda(\pi) = \pi$ for the membership problem, and $\lambda(\pi) = \|\pi\|$ for the counting problem. For any algorithm solving these kind of problems, the information available for a node to infer $\lambda(\pi)$ includes the channel feedback history $r(\pi)$ and the local initial state $\pi[i]$. Lemma 2.2 asserts that the inference of $\lambda(\pi)$ in any deterministic algorithm $\mathcal{A}$ must solely rely on analyzing the channel feedback $r_{\mathcal{A}}(\pi)$, and the knowledge of $\pi[i]$ cannot be effectively utilized by any node in the inference. The proof of Lemma 2.2 is based on Lemma 2.1.

Lemma 2.2. *In the beeping model, for any deterministic algorithm $\mathcal{A}$ that lets every node learn $\lambda(\pi)$, let π and π' be two instances of the problem to be solved, if $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi')$, then $\lambda(\pi) = \lambda(\pi')$.*

Proof. For contradiction, suppose $\lambda(\pi) \neq \lambda(\pi \vee \pi')$ and $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi')$. We know $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi \vee \pi')$ by Lemma 2.1. In addition, there exists i^* for which $\pi[i^*] = (\pi \vee \pi')[i^*]$. So, all information for node i^* to distinguish the two different values of $\lambda(\pi)$ and $\lambda(\pi \vee \pi')$ is the same, which means it cannot distinguish them. Thus, the algorithm $\mathcal{A}$ fails to let *every* node learn $\lambda(\pi)$. This contradiction implies that any algorithm must have $\lambda(\pi) = \lambda(\pi \vee \pi')$ when $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi')$. In the same way we also get $\lambda(\pi') = \lambda(\pi \vee \pi')$, so $\lambda(\pi) = \lambda(\pi')$. □

Lemma 2.2 implies that the number of different channel feedback histories generated by a deterministic algorithm computing $\lambda(\pi)$ is no less than the number of possible values of $\lambda(\pi)$. For the membership problem, there are $\sum_{i=0}^{k}\binom{n}{i}$ different values of $\lambda(\pi)$, so we have a lower bound of $\Omega(\log\binom{n}{k}) = \Omega(k\log\frac{n}{k})$ for any algorithm that correctly solves it. For the counting problem, however, two different instances may have the same number of "1" (i.e., $\lambda(\pi) = \lambda(\pi')$ for some $\pi \neq \pi'$), so one might expect an efficient algorithm that beats the lower bound of $\Omega(k\log\frac{n}{k})$ by "sharing" the same channel feedback history between different instances. Theorem 2.3 proves that this is impossible: In the beeping model, it is not easier to count the number of active nodes than to identify them all.

Theorem 2.3. *For any deterministic algorithm $\mathcal{A}$ and any problem instance π with $\|\pi\| > 1$ (i.e. $k > 1$), if $\mathcal{A}$ lets every node learn $\|\pi\|$ with channel feedback history $r_{\mathcal{A}}(\pi)$, then we can construct an algorithm $\mathcal{A}'$ that lets every node learn π with exactly the same channel feedback history $r_{\mathcal{A}}(\pi)$.*

Proof. We prove that the mapping from π to $r_{\mathcal{A}}(\pi)$ for any deterministic counting algorithm $\mathcal{A}$ must be injective, i.e., if $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi')$ then $\pi = \pi'$. Since every instance π has a different channel feedback history $r_{\mathcal{A}}(\pi)$ when $\mathcal{A}$ terminates, $\mathcal{A}'$ simply remembers the entire table of the one-to-one mapping from $r_{\mathcal{A}}(\pi)$ to π, thus solving the membership problem once $r_{\mathcal{A}}(\pi)$ is given.

The proof of injectivity is by contradiction. Since $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi')$, by Lemma 2.2 we have $\|\pi\| = \|\pi'\|$. Suppose $\pi \neq \pi'$, then there must be some i^* with $\pi[i^*] = 0$ and $\pi'[i^*] = 1$, so we have $\|\pi \vee \pi'\| > \|\pi\|$. On the other hand, since $r_{\mathcal{A}}(\pi) = r_{\mathcal{A}}(\pi')$, by Lemma 2.1 we have $r_{\mathcal{A}}(\pi \vee \pi') = r_{\mathcal{A}}(\pi)$, and then by Lemma 2.2 we have $\|\pi \vee \pi'\| = \|\pi\|$, a contradiction. □

We remark that Theorem 2.3 can be naturally generalized to prove the equivalence with the membership problem for more problems defined by $\lambda(\pi)$ in the beeping model. Actually, any deterministic algorithm solving a problem defined by $\lambda(\pi)$ can be used to solve the membership problem with the same channel feedback history as long as the function $\lambda(\cdot)$ has the property that, for any two different instances π and π', $\lambda(\pi) = \lambda(\pi \vee \pi') \Rightarrow \lambda(\pi) \neq \lambda(\pi')$.

Moreover, for problems that cannot be directly represented by a function of π, we may still prove their equivalence with MP by proving that an algorithm for this problem can solve the counting problem. This allows us to prove the main theorem of this section.

Theorem 2.4. *For any problem instance π with $\|\pi\| > 1$ (i.e. $k > 1$) and any positive integer T, if any deterministic algorithm $\mathcal{A}$ solves conflict resolution under π in T time slots, then we can construct an algorithm $\mathcal{A}'$ that lets every node learn π in exactly T time slots.*

Proof. The idea is to construct a counting algorithm $\tilde{\mathcal{A}}$ from the conflict resolution algorithm $\mathcal{A}$. Recall that a time slot is "exclusively-used" if only one single node is allowed to beep in that time slot. Given a deterministic conflict resolution algorithm $\mathcal{A}$, the algorithm $\tilde{\mathcal{A}}$ runs $\mathcal{A}$, and lets each active node beep in the first exclusively-used time slot it has and keep quiet since then. When $\mathcal{A}$ terminates, $\tilde{\mathcal{A}}$ lets each node count the number of beep signals in all the exclusively-used time slots, which is also the number of active nodes. So $\tilde{\mathcal{A}}$ is a valid counting algorithm (and $|r_{\tilde{\mathcal{A}}}(\pi)| = T$). By Theorem 2.3 we know that the information of $r_{\tilde{\mathcal{A}}}(\pi)$ is already enough to solve the membership problem. □

Theorem 2.4 shows that MP can be reduced to CR at no additional cost in time-slots. Reversely, the reduction from CR to MP requires k additional time-slots. Since any CR algorithm needs at least k time-slots to let every node transmit successfully once, the reduction to the MP does not change the performance bound for any deterministic CR algorithm. Thus, the two problems share the same upper and lower bounds in beeping model. As mentioned before, a simple counting argument gives the lower bound $\Omega(k\log\frac{n}{k})$ for the membership problem, which therefore also applies to conflict resolution. Furthermore, Theorem 2.5 shows that the same lower bound also applies to any *randomized* algorithm that solves either problem of CR and MP with constant probability.

Theorem 2.5. *In the Beeping Model with single channel, for any constant* $0 \leq \lambda < 1$, *every randomized algorithm requires at least* $\Omega(k \log \frac{n}{k})$ *time slots to solve Conflict Resolution or the Membership Problem with success probability* λ *under the worst-case distribution of problem instances.*

Proof. By the so-called *entropy argument*, we prove that any randomized algorithm that always correctly solves the Membership Problem when terminating (i.e. Las Vegas algorithms) has the expected run time of $\Omega(k \log \frac{n}{k})$. Specifically, due to Yao's principle, a distributed Las Vegas algorithm $\mathcal{A}$ is a stochastic distribution over a set of correct deterministic algorithms, where each deterministic algorithm $\mathcal{A}'$ in this set owns a different table $R_{\mathcal{A}'} : [n] \to \{0,1\}^*$ and simulates $\mathcal{A}$ by using the *fixed* sequence $R_{\mathcal{A}'}(i)$ to replace the random numbers used in node i. On the other hand, due to Shannon's encoding theorem, no deterministic membership algorithm can have an expected performance better than $\log_2 \binom{n}{k} = \Omega(k \log \frac{n}{k})$ under the uniform distribution over the $\binom{n}{k}$ different problem instances, thus any stochastic distribution over any subset of deterministic membership algorithms, i.e., any Las Vegas membership algorithm, must need $\Omega(k \log \frac{n}{k})$ time slots under the uniform distribution over problem instances.

Due to Theorem 2.4, every deterministic conflict resolution algorithm must have an average performance of $\Omega(k \log \frac{n}{k})$ (for otherwise we will find a deterministic membership algorithm beating this bound), and thus every Las Vegas conflict resolution algorithm also has a worst-case performance of $\Omega(k \log \frac{n}{k})$.

Finally, for any randomized algorithm $\mathcal{A}$ that solves either the membership problem or conflict resolution with constant success probability $0 < \lambda < 1$ in T time slots (under the worst-case distribution of problem instances), we can verify its correctness in one single time slot by letting active nodes to report whether anyone is missing, and thus can construct a Las Vegas algorithm by repeatedly running $\mathcal{A}$ until it is correct. The Las Vegas algorithm thus constructed has the expected running time of $\sum_{i=1}^{\infty}(1-\lambda)^{i-1}\lambda \cdot i \cdot T = T/\lambda$. As proved above,any Las Vegas algorithm for either problem has expected running time of $\Omega(k \log \frac{n}{k})$ under the uniform distribution of problem instances, so $T = \Omega(k \log \frac{n}{k})$. □

3 Efficient Algorithm in Beeping Model with Parallel Channels

In last section we proved a lower bound of $\Omega(k \log \frac{n}{k})$ for both MP and CR when only one single beeping channel is available. In this section we give an efficient algorithm for the parallel case. We call our algorithm *the Funnel Algorithm.* The algorithm has a time complexity of $O(\log k \log \log_k n)$, yet only uses $O(\max\{k \log_k n, k^2\})$ parallel channels. As we have already shown that MP and CR are equally hard problems, we describe our algorithms in the context of the Membership Problem.

3.1 The Funnel Algorithm

The algorithm runs in a sequence of *iterations*, and the idea is to gradually reduce the problem size n by renaming the active nodes to the name space $\{0, ..., k^{d_t} - 1\}$ in iteration t. The values $d_1, d_2, \ldots$ decrease gradually during the iteration and the algorithm terminates when $d_t = 1$, at that time each active node has an ID $i' \in \{0, ..., k-1\}$, and then each node locally recovers the original ID's of all active nodes based on the channel feedback history. The sequence $D = \{d_1, d_2, ...\}$ is called an *iteration policy*, which has significant impact on the algorithm's performance. We now present the general iteration framework first, and then define the concrete iteration policy we use to achieve our results.

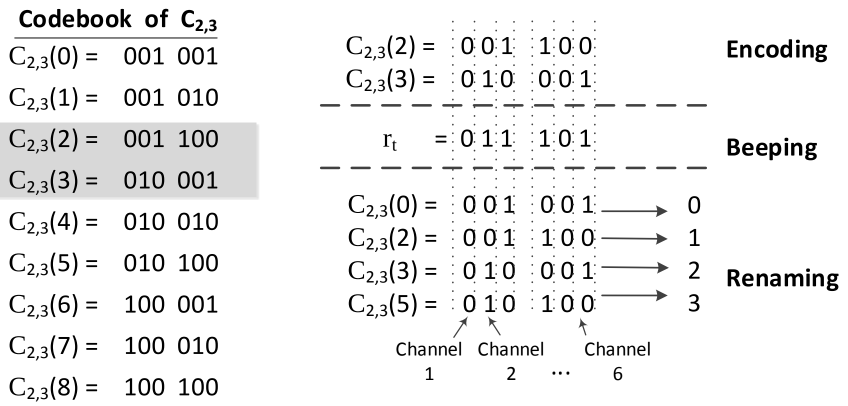

Figure 2: Illustration of the encoding-beeping-renaming procedure when n_t = 9, d_t = 2, $q_t = 3$. The left side is the codebook of the $(2, 3)$-identity code. The right side illustrates how the encoding/beeping/renaming subroutines work in the case that nodes 2 and 3 are active.

Iteration Framework: Given an iteration policy $D = \{d_1, d_2, ...\}$, in each iteration t, the algorithm works by running the following encoding-beeping-renaming procedure at each active node (see Figure 2 for an illustration):

- *Encoding*. Suppose the name space is $\{0, ..., n_t - 1\}$ in iteration t. Let $q_t = \lceil n_t^{\frac{1}{d_t}} \rceil$, each node locally encodes its current ID u_t with the so-called (d_t, q_t)-*identity code*. The (d, q)-identity code encodes an integer $u \in \{0, \dots, q^d - 1\}$ into a bit vector $C_{d,q}(u)$ by first transforming u into the q-nary format $[u]_q = (\mu_1, ..., \mu_d)$, then encoding each μ_j with a q-bit vector that is 1 in the μ_j'th position and "0" everywhere else. For example, $C_{2,3}(2) = C_{2,3}\big((0, 2)\big) = 001\ 100$, and $C_{2,3}(3) = C_{2,3}\big((1, 0)\big) = 010\ 001$. The code length of $C_{d_t,q_t}(u_t)$ is denoted by $L(n_t, d_t)$, where

$$L(n_t, d_t) = d_t \cdot q_t = d_t \lceil n_t^{1/d_t} \rceil < d_t n_t^{1/d_t} + d_t. \tag{4}$$

- *Beeping*. Each active node "beeps out" all bits of its codeword $C_{d_t,q_t}(u_t)$ in parallel, using $L(n_t, d_t)$ single-channel accesses. This takes one time slot. Since all active nodes beep simultaneously, the resulting channel feedback $r_t = (\gamma_1, ..., \gamma_{d_t})$ is the bitwise OR of all these codewords. Note that each segment γ_j in r_t is a q_t-bit vector that may have multiple "1"s in it.
- *Renaming*. Given channel feedback r_t, the "possible" codewords of any active node u_t can only be a combination of the "1"s in *different* segments of r_t. For example (see Figure 2), for the $(2, 3)$-code, the channel feedback $r_t = 011\ 101$ can only be generated by a subset of the nodes $\{0, 2, 3, 5\}$. The number of these "possible" codewords is $\prod_j \|\gamma_j\| = k^{d_t}$ in the worst case. So each active node can locally rename itself with a new unique ID $u_{t+1} \in \{0, ..., k^{d_t} - 1\}$, where u_{t+1} is the order of its original ID u_t among all the (at most k^{d_t}) possible ID's. For example, for the node with $u_t = 3$, there is two possible ID's smaller than u_t (i.e., 0 and 2), so $u_{t+1} = 2$.

Note that a node can locally recover $C_{d_t,q_t}(u_t)$ for *any* given u_{t+1} (u_{t+1} doesn't necessarily belong to the node itself) as long as d_t, q_t, and r_t are known. With $C_{d_t,q_t}(u_t)$, the node can further recover u_t locally. Therefore, when the algorithm terminates with $d_T = 1$, every node can locally recover the original identifier of each active node by sequentially recover $u_T, u_{T-1}, \ldots$, down to until u_1.

Iteration Policy: Any iteration policy ending with $d_T = 1$ and having $d_{t+1} \leq d_t$ for any $t < T$ returns the correct answer to the problem. Among them, we choose the iteration policy D^* for performance consideration.

Definition 3.1. *Let T be the maximal integer satisfying $(1 + \frac{1}{\ln k})^T \leq \log_k n$. Define $\tilde{d}_t = (1 + \frac{1}{\ln k})^{T-t}$ for $t \geq 0$. The iteration policy $D^* = \{d_1, ..., d_T\}$, where*

$$d_t = \begin{cases} \lceil \tilde{d}_t \rceil & \textit{when } \tilde{d}_t \geq \ln k \\ d_{t-1} - 1 & \textit{when } \tilde{d}_t < \ln k \end{cases}. \tag{5}$$

3.2 Performance Analysis

In the Funnel Algorithm, each node makes $L(n_t, d_t)$ single-channel accesses in each iteration t, which takes one single time slot when $L(n_t, d_t)$ number of channels are available. Therefore, the algorithm's time complexity corresponds to the number of iterations, and the number of channels it requires corresponds to the maximal number of single-channel accesses in all iterations, i.e., $\max_t L(n_t, d_t)$. In the following, we first prove bounds for the time slots the Funnel Algorithm needs (Theorem 3.1), then characterizes its efficiency in channel usage and computational complexity (Theorem 3.3).

Theorem 3.1. *The Funnel Algorithm takes $O(\log k \log \log_k n)$ time slots to terminate in the worst case.*

Proof. Clearly the algorithm terminates in one time slo twhen $k = 1$ (i.e. at most one node is active). For $k \geq 2$, by Definition 3.1, the Funnel Algorithm terminates in no more than $T + \ln k + 1$ time slots in any case, and we have $((1 + \frac{1}{\ln k})^{\ln k})^{T/\ln k} \leq \log_k n$, which means

$$T \leq \ln k \cdot \ln \log_k n / \ln \left((1 + \frac{1}{\ln k})^{\ln k} \right). \tag{6}$$

Since $1.85 < (1 + \frac{1}{\ln k})^{\ln k} < e$ for $k \geq 2$, we have $T \leq 1.63 \cdot \ln k \cdot \ln \log_k n$. □

Lemma 3.2. *The Funnel Algorithm makes $O(k \log \frac{n}{k} + k^2)$ single-channel accesses in the worst case, that is, $\sum_{t=1}^{T} L(n_t, d_t) = O(k \log \frac{n}{k} + k^2)$.*

Proof. To prove the bound on the total number of channel accesses, we show that the iterations with $\tilde{d}_t \geq \ln k$ collectively have $O(k \log \frac{n}{k})$ channel-accesses, and the remaining iterations (i.e. with $\tilde{d}_t < \ln k$) collectively have $\Theta(k^2)$ channel-accesses, thus the sum of channel accesses over all iterations is $O(k \log \frac{n}{k} + k^2)$.

When $\tilde{d}_t \geq \ln k$: Since $d_t = \lceil \tilde{d}_t \rceil$, we have $\tilde{d}_t \leq d_t \leq \tilde{d}_t + 1$ and $1 \leq \frac{\tilde{d}_t}{\ln k} \leq \frac{d_t}{\ln k}$. By Definition 3.1 we also have $(1 + \frac{1}{\ln k})\tilde{d}_t = \tilde{d}_{t-1}$. Combining these results together yields a chain of inequalities

$$\tilde{d}_t \;\leq\; d_t \;\leq\; (1 + \frac{1}{\ln k})\tilde{d}_t \;=\; \tilde{d}_{t-1} \;\leq\; d_{t-1} \;\leq\; (1 + \frac{2}{\ln k})\tilde{d}_t. \tag{7}$$

We know by Eq.(4) that the number of single-channel accesses made in iteration t is less than $d_t(k^{\frac{d_{t-1}}{d_t}} + 1)$. Then, by (7), we have

$$d_t(k^{\frac{d_{t-1}}{d_t}} + 1) \leq (1 + \frac{1}{\ln k})\tilde{d}_t(k^{\frac{(1+\frac{2}{\ln k})\tilde{d}_t}{d_t}} + 1) \leq (1 + \frac{1}{\ln 2})\tilde{d}_t(e^2 k + 1) \leq 2.45(e^2 k + 1)\tilde{d}_t.$$

Let t^* be the last iteration with $\tilde{d}_t \geq \ln k$. We know that the total number of single-channel accesses made from iteration 1 through iteration t^* is less than

$$\begin{aligned}
&\sum_{1}^{t^*} d_t(k^{\frac{d_{t-1}}{d_t}}+1) \;\leq\; \sum_{1}^{T} d_t(k^{\frac{d_{t-1}}{d_t}}+1) \;\leq\; \sum_{1}^{T} 2.45(e^2k+1)\tilde{d}_t \\
&= \; 2.45(e^2k+1)\sum_{0}^{T-1}(1+\frac{1}{\ln k})^t \;=\; 2.45(e^2k+1)\ln k((1+\frac{1}{\ln k})^T-1) \\
&\leq \; 2.45(e^2k+1)\ln k(\log_k n-1) \;=\; 2.45(e^2k+1)\ln\frac{n}{k} \;=\; O(k\log\frac{n}{k}).
\end{aligned}$$

When $\tilde{d}_t < \ln k$: Let t^* be the first iteration with $\tilde{d}_t < \ln k$, we have $d_{t^*} = d_{t^*-1}-1$ and $d_{t^*-1} = \lceil \tilde{d}_{t^*-1}\rceil = \lceil \tilde{d}_{t^*}(1+\frac{1}{\ln k})\rceil < \lceil \ln k(1+\frac{1}{\ln k})\rceil$, and thus $d_{t^*} < \lceil \ln k+1\rceil - 1 \leq \lfloor \ln k\rfloor + 1$. Then we know that the number of single-channel accesses used in all iterations with $\tilde{d}_t < \ln k$ is

$$\begin{aligned}
\sum_{\{t:\tilde{d}_t<\ln k\}} L(n_t,d_t) &= \sum_{d=1}^{\lfloor \ln k\rfloor+1} d(k^{\frac{d+1}{d}}+1) = \sum_{d=1}^{\lfloor \ln k\rfloor} dk^{\frac{d+1}{d}} + O(k\log k) + O(\log^2 k) \\
&= k^2 + 2k^{\frac{3}{2}} + k\sum_{d=3}^{\lfloor \ln k\rfloor} dk^{\frac{1}{d}} + O(k\log k) + O(\log^2 k)
\end{aligned}$$

Now it is sufficient to prove that $\sum_{d=3}^{\lfloor \ln k\rfloor} dk^{\frac{1}{d}} = o(k)$. By using Euler's Approximation,

$$\begin{aligned}
\sum_{d=3}^{\lfloor \ln k\rfloor} dk^{\frac{1}{d}} < \int_2^{\ln k} xk^{\frac{1}{x}}\mathrm{d}x \;&=\; \frac{1}{2}xk^{\frac{1}{x}}(\ln k+x) - \frac{1}{2}(\ln k)^2\mathrm{Ei}(\frac{\ln k}{x})\Big|_2^{\ln k} \\
&= \frac{1}{2}(\ln k)^2\mathrm{Ei}(\frac{\ln k}{2}) - \sqrt{k}(\ln k+2) + (e-\frac{\mathrm{Ei}(1)}{2})(\ln k)^2,
\end{aligned} \tag{8}$$

where $\mathrm{Ei}(x)$ is the *Exponential Integral* function defined as $\mathrm{Ei}(x) = \int_{-\infty}^{x}\frac{e^t}{t}\mathrm{d}t$, which is known to have no closed form expression. However, by noticing that $\frac{e^t}{t}$ is monotonically increasing for $t > 1$, we can derive an upper bound of $\mathrm{Ei}(x)$:

$$\mathrm{Ei}(x) \;=\; \int_{-\infty}^{x}\frac{e^t}{t}\mathrm{d}t \;=\; \mathrm{Ei}(1) + \int_1^x \frac{e^t}{t}\mathrm{d}t \;<\; \mathrm{Ei}(1) + \frac{e^x}{x}(x-1). \tag{9}$$

Substituting Eq.(9) back to Eq.(8), we get

$$\sum_{d=3}^{\lfloor \ln k\rfloor} dk^{\frac{1}{d}} \;<\; \frac{1}{2}(\ln k)^2\frac{2e^{\frac{\ln k}{2}}}{\ln k}(\frac{\ln k}{2}-1) - \sqrt{k}(\ln k+2) + e(\ln k)^2 \;=\; \Theta\Big(\sqrt{k}(\ln k)^2\Big).$$

$\square$

Theorem 3.3. *The Funnel Algorithm uses $O(k\log_k n + k^2)$ parallel channels and has computational complexity of $O(k^2\log\frac{n}{k} + k^3)$.*

Proof. We first prove upper bound about channel usage, i.e., $\max_{t\in[T]} L(n_t, d_t) = O(k \log_k n + k^2)$. Recall that we already proved in Lemma 3.2 that $\sum_t L(n_t, d_t) = \Theta(k^2)$ for the iterations with $\tilde{d}_t < \ln k$, which means $L(n_t, d_t) = O(k^2)$ for any t with $\tilde{d}_t < \ln k$. For iterations with $\tilde{d}_t \geq \ln k$, we have

$$\max_t L(n_t, d_t) \leq \max_t d_t(n_t^{\frac{1}{d_t}} + 1) = \max_t d_t(k^{\frac{d_{t-1}}{d_t}} + 1) \tag{10}$$

From Inequality (7) we know $\frac{d_{t-1}}{d_t} \leq 1 + \frac{2}{\ln k}$. Substituting this to Eq. (10) yields

$$\max_t L(n_t, d_t) \leq d_t(k^{1+\frac{2}{\ln k}} + 1) = d_t(k \cdot e^2 + 1).$$

By Definition 3.1 we have $d_t \leq \log_k n$ for any t, which concludes with $L(n_t, d_t) = O(k \log_k n)$ for any t with $\tilde{d}_t \geq \ln k$.

Finally, since the local computation of every subroutine (encoding/ beeping/ renaming/ decoding) in the Funnel Algorithm is linear to the code length $L(n_t, d_t)$, the computational complexity for the Funnel Algorithm to recover k identifiers is $O(k \cdot S)$, where $S = \sum_t L(n_t, d_t)$ is the total number of single-channel accesses. Then by Lemma 3.2, the Funnel Algorithm has computational complexity of $O(k \cdot S) = O(k^2 \log \frac{n}{k} + k^3)$. $\square$

3.3 Minimizing Total Channel Accesses and the Integrality Gap

In this subsection we present some interesting observations about the *most economic* policy that minimizes the total number of channel accesses under the general framework of the Funnel Algorithm. Specifically, we show that the underlying *continuous dynamic programming* equations of the Funnel algorithm does match the information-theoretical lower bound of the problem, but there turns out to be an *integrality gap* caused by the discrete numbers of encoding dimensions that prevents the Funnel algorithm to achieve this idealized optimum.

Lemma 3.4. *The total number of channel accesses used by the optimal iteration policy of the Funnel Algorithm is*

$$f_k(n) = \min_{1\leq d\leq log_k n} \{d\lceil n^{\frac{1}{d}}\rceil + f_k(k^d)\} \qquad n, k, d \in \mathbb{Z}^+, \;\; f_k(k) = 0. \tag{11}$$

Proof. Let $f_k(n)$ be the number of single-channel accesses the optimal policy needs before terminating when the size of the initial candidate list is n. Since the Funnel Algorithm terminates when $n_t = k$, we have $f_k(k) = 0$. In each iteration t, the algorithm takes $d_t\lceil n_t^{1/d_t}\rceil$ single-channel accesses for each node to send the codeword and reduces the size of the candidate list to k^{d_t}, so it needs $d_t\lceil n_t^{1/d_t}\rceil + f_k(k^{d_t})$ single-channel accesses in total to terminate. Finally, $d \leq \log_k n$ is for making sure $k^d \leq n$. $\square$

Note that all n's in the recursion (11) are powers of k except the first one (which is the raw input). This implies that, to compute the optimal policy we only need to compute a lookup table $NEXT(d_t) = d_{t+1}$, indexed by d (rather than indexed by n). The table can be filled out in local time $O(\log_k^2 n)$ using dynamic programming. Interestingly, the following Theorem shows that the iteration policy of Definition 3.1 is within the same asymptotic bound with the optimal iteration policy presented in Lemma 3.4, while the closed-form computation of this iteration policy is much faster than computing the optimal policy by dynamic programming.

Theorem 3.5. *Definition 3.1 is asymptotically optimal among all valid deterministic iteration policies of the Funnel Algorithm in the number of channel accesses.*

Proof. The last iteration of any iteration policy uses at least k^2 single-channel accesses. On the other hand, we know that there is a lower bound of $\Omega(k \log \frac{n}{k})$ for any deterministic algorithm. This means a lower bound of $\Omega(\max\{k \log \frac{n}{k}, k^2\}) = \Omega(k \log \frac{n}{k} + k^2)$ for any deterministic iteration policy, which is already achieved by the iteration policy D^*, due to Theorem 3.1. □

We know that the optimal sequential time complexity for an algorithm solving MP in the beeping model is $\Theta(k \log \frac{n}{k})$, and yet the previous theorem shows that even the best possible iteration schedule of the Funnel Algorithm cannot achieve this optimal bound, but instead "only" achieves a bound of $O(k \log \frac{n}{k} + k^2)$. The question is why this is the case. Interestingly, we can prove that this gap is due to an inherent *integrality gap*: If the iteration schedule used by the Funnel Algorithm were allowed to use fractional values for the d_t of different iterations, then the Funnel Algorithm would be able to achieve the optimal bound. This reveals that the suboptimality of the Funnel Algorithm is solely due to this integrality gap. The proof is by analytically solving a non-trivial dynamic programming equation.

Theorem 3.6. *There exists a fractional iteration schedule that is allowed to use real values for d_t and uses only $O(k \log \frac{n}{k})$ single-channel accesses.*

Proof. The proof is by analytically solving the following *dynamic programming equation*. Actually we show that, for $y \in \mathbb{R}$, $y \geq 1$, $\tau'(x) \in \mathbb{R}$, $f'_k(x) \in \mathbb{R}$, the solution of

$$f'_k(y) = \min_{1 \leq \tau' < \log_k y} \{\tau' \cdot y^{\frac{1}{\tau'}} + f'_k(k^{\tau'})\} \quad , \ f'_k(k) = 0 \tag{12}$$

is

$$\begin{aligned} \log_k y &= (1 + \frac{1}{\ln k})\tau' \\ f'_k(y) &= ek \ln \frac{y}{k}. \end{aligned} \tag{13}$$

By defining $y = k^x$ and $g_k(x) = f'_k(k^x)$, we rewrite Eq.(12) to Eq.(14), and prove that, for $x \in \mathbb{R}$, $x \geq 1$, $\tau(x) \in \mathbb{R}$, $g_k(x) \in \mathbb{R}$, the solution of

$$g_k(x) = \min_{1 \leq \tau < x} \{\tau \cdot k^{\frac{x}{\tau}} + g_k(\tau)\} \quad , \ g_k(1) = 0 \tag{14}$$

is

$$\begin{aligned} x &= (1 + \frac{1}{\ln k})\tau \\ g_k(x) &= (e \cdot k \cdot \ln k)(x - 1), \end{aligned} \tag{15}$$

which yields

$$\begin{aligned} \log_k y &= (1 + \frac{1}{\ln k})\tau' \\ f'_k(y) &= g_k(\log_k y) = (ek \ln k)\frac{\ln y}{\ln k} - ek \ln k = ek \ln \frac{y}{k}. \end{aligned} \tag{16}$$

To solve Eq. (14), observe that Eq. (14) can be rewritten as

$$g_k(x) = \tau k^{\frac{x}{\tau}} + g_k(\tau) \tag{17}$$

$$\frac{\partial}{\partial \tau}(\tau k^{\frac{x}{\tau}} + g_k(\tau)) = 0. \tag{18}$$

For undetermined coefficient c and b, let $\tau = \frac{x}{c}$ and $g_k(x) = \frac{k^c}{c-1}x + b$, we first verify Eq.(17) as follows:

$$\tau k^{\frac{x}{\tau}} + g_k(\tau) = \frac{x}{c}k^c + \frac{k^c}{c-1}\frac{x}{c} + b = g_k(x).$$

The parameter c can be determined by substituting $\tau = \frac{x}{c}$ and $g_k(x) = \frac{k^c}{c-1}x + b$ into Eq.(18) as follows:

$$\begin{aligned}
&\frac{\partial}{\partial \tau}\tau k^{\frac{x}{\tau}} + \frac{\mathrm{d}}{\mathrm{d}\tau}g_k(\tau) = 0\\
&k^{\frac{x}{\tau}}(1 - \ln k \cdot \frac{x}{\tau}) + \frac{\mathrm{d}}{\mathrm{d}\tau}g_k(\tau) = 0\\
&k^c(1 - c \cdot \ln k) + \frac{k^c}{c-1} = 0\\
&c = 1 + \frac{1}{\ln k}.
\end{aligned} \tag{19}$$

Further substituting Equation (19) back to $g_k(x) = \frac{k^c}{c-1}x + b$ yields

$$g_k(x) = \ln k \cdot k \cdot k^{\frac{1}{\ln k}} \cdot x + b = \ln k \cdot k \cdot e \cdot x + b.$$

The parameter b can be determined by considering the initial condition $g_k(1) = 0$, then we get

$$g_k(x) = (e \cdot k \cdot \ln k)(x - 1).$$

□

3.4 Crash Tolerance

In addition to its efficiency, the Funnel Algorithm is also *resilient to crash failures (fail-stops)*, in the sense that the algorithm guarantees to identify all active nodes that remain alive (not crashed) when the algorithm terminates, assuming an adversary crashes an arbitrary subset of nodes in any time slot. This is because the Funnel Algorithm maintains a candidate list until the last iteration, without making any irreversible decisions regarding the activeness of any node in the list (=no false positives). Also, the crash of an active node never leads to the removal of any other active node from the candidate list (=no false negatives).

Theorem 3.7. *Let $N = \{1, ..., n\}$, and let $A_t \subseteq N$ be the active set at the end of time slot t for any $t > 0$ and A_0 be the active set when the Funnel Algorithm starts. For any infinite sequence $A_0, A_1, A_2, ...$ satisfying $A_t \subseteq A_{t-1}$ for any $t > 0$, the Funnel Algorithm terminates no later than in the case of given sequence $\{A_t = A_0$ for any $t > 0\}$, and returns A_T if it terminates at time slot T.*

4 Related Work

The study of Conflict Resolution in the traditional ternary-feedback model at least dates back to the late 70's, where several *adaptive* solutions were found [10] [4] [19] [17]. In general, these solutions fall into two categories [20]: the *reservation protocols* [10], which precisely reduce Conflict Resolution to Membership Problem; and the *direct transmission protocols* [4] [19] [17], where active nodes directly send and re-send their messages in a coordinated way, without first identifying all channel competitors. While there is the *entropy lower bound* of $\log_3 \binom{n}{k} = \Omega(k \log \frac{n}{k})$ for any reservation protocol (i.e. Membership-based

CR algorithm) [10], randomized direct transmission protocols can correctly solve the problem of CR with expected time $\Theta(k)$ in any case [9]. These facts show a separation in hardness between the two problems of MP and CR in the ternary-feedback model. On the other hand, Greenberg and Winograd [9] proved a lower bound of $\Omega(k \log_k n)$ for any deterministic conflict resolution algorithm, establishing another seperation between deterministic algorithms and randomized algorithms for the problem of CR.

Interestingly, Komlos and Greenberg [13] demonstrated the existence of a special type of adaptive CR algorithms which achieves $O(k \log \frac{n}{k})$ in time and has the nice feature that only the feedback about whether a node *itself* succeeds in sending is needed (thus cannot be applied in beeping model). This feature turns out to be very useful in networks with unknown topology [5]. However, such an algorithm, although does employ non-adaptive conflict resolution *schemes*, still has to adjust its behaviors depending on real-time channel feedbacks, thus cannot be parallelized significantly.

Many studies on radio networks assume another multiple access model, called *radio network model without collision detection*, where nodes are unable to distinguish collisions (the conflict state) from the background noise (the idle state). Although also assuming binary feedbacks, the radio-network model returns whether a time slot is a success (in contrast, the beeping model returns whether a time slot is idle). Interestingly, we can also observe in the radio-network model the separation in hardness between the problems of CR and MP, as well as the separation in efficiency between deterministic and randomized CR algorithms [3] – both disappear in the beeping model.

Jack Wolf and his collaborators [20] was the first to identify the close connection between Conflict Resolution/ Membership Problem and *Combinatorial Group Testing* (CGT), a classic combinatorial searching problem that asks to efficiently identify the at most k *positives* from a large population of size n by repeated querying whether a chosen subset contains at least one positive. CGT algorithms apply for both the traditional ternary-feedback model and the beeping model. Applying *fully-adaptive* CGT algorithms [7] can solve Membership Problem in $O(k \log \frac{n}{k})$ time with single channel, but still in $O(k \log \frac{n}{k})$ time if multiple channels are available. On the other hand, *non-adaptive* [4] [12] [18] [11] and *partial-adaptive* [3] CGT algorithms can be used to solve Membership Problem in $O(1)$ time slot with $O(k^2 \log n)$ and $O(k \log n)$ channels, respectively. These works utilize sophisticated coding techniques (e.g. list-decoding code), which either need a $O(n)$ "decoding" time [12] [18], or only provide randomized solutions [11] [3] (though some can be derandomized in certain conditions).

5 Conclusion

We have proved two new results on the distributed complexity of computation in Beeping Model. We present a new algorithm that improves upon the best-known existing solutions; and we show that two key problems in this model are equally hard. The latter result, in particular, sheds new light not only on the Beeping Model itself, but also on its relationship to the most well-studied model for medium access channels: the collision-detection model with ternary feedback. Our results suggest that even for such basic problems such as conflict resolution and the membership problem, the Beeping Model may behave fundamentally differently with other classic channel models. In particular, it seems that the inability to detect successful

[3]In the radio-network model, there exist $O(k + \log n)$ randomized CR algorithms [16], while there is the entropy lower bound of $\Omega(k \log \frac{n}{k})$ for randomized MP algorithms and a lower bound of $\Omega(k \log \frac{n}{k})$ for deterministic CR algorithms [5].

[4]It may be sometimes confusing to compare the capability in parallelism between *non-adaptive group testing scheme* and *non-adaptive conflict resolution scheme*. The former guarantees a group testing algorithm that runs fully in parallel, while the latter may leads to sequential conflict resolution algorithms in general. This is because an individual in group testing will always return the same result (positive or negative) whenever it is queried, but a node in conflict resolution has the additional flexibility to *decide to not transmit* even in time slots when it is allowed to transmit.

communications (partially due to the lack of sophisticated modulation and coding schemes) has made the beeping model quite different from traditional models. As future works, it may be interesting to further investigate the relationship between these (and other) channel access models.

Acknowledgements. The authors thank Dongxiao Yu and Wei Chen for the inspiring discussions about this work. We also thank the anonymous reviewers who gave valuable comments on early versions of this paper.

References

[1] Yehuda Afek, Noga Alon, Ziv Bar-Joseph, Alejandro Cornejo, Bernhard Haeupler, and Fabian Kuhn. Beeping a maximal independent set. In *Proceedings of the 25th international conference on Distributed computing*, DISC'11, 2011.

[2] Yehuda Afek, Noga Alon, Omer Barad, Eran Hornstein, Naama Barkai, and Ziv Bar-Joseph. A biological solution to a fundamental distributed computing problem. *Science*, 331(6014):183–185, 2011.

[3] Annalisa De Bonis, Leszek Gasieniec, and Ugo Vaccaro. Generalized framework for selectors with applications in optimal group testing. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming*, ICALP '03, 2003.

[4] J. Capetanakis. Tree algorithms for packet broadcast channels. *Information Theory, IEEE Transactions on*, 25(5):505 – 515, sep 1979.

[5] Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA '01, 2001.

[6] Alejandro Cornejo and Fabian Kuhn. Deploying wireless networks with beeps. In *Proceedings of the 24th international conference on Distributed computing*, DISC'10, 2010.

[7] Dingzhu Du and Frank Hwang. *Combinatorial group testing and its applications*. World Scientific, 2000.

[8] Sylvain Gelly, Levente Kocsis, Marc Schoenauer, Michèle Sebag, David Silver, Csaba Szepesvári, and Olivier Teytaud. The grand challenge of computer go: Monte carlo tree search and extensions. *Commun. ACM*, 55(3):106–113, March 2012.

[9] Albert G. Greenberg and Schmuel Winograd. A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *J. ACM*, 32(3), July 1985.

[10] J. Hayes. An adaptive technique for local distribution. *Communications, IEEE Transactions on*, 26(8):1178 – 1186, aug 1978.

[11] Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, 2010.

[12] W. Kautz and R. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10, 1964.

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

[14] Dariusz R. Kowalski. On selection problem in radio networks. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, PODC '05, 2005.

[15] Chou Hsiung Li. A sequential method for screening experimental variables. *Journal of the American Statistical Association*, 57(298):455–477, 1962.

[16] C. U. Martel. Maximum finding on a multiple access broadcast network. *Information Processing Letters*, 52 (1994) 7 - 13.

[17] J. L. Massey. Collision-resolution algorithms and random-access communications. *Technical Report UCLA-ENG-8016*, April 1980.

[18] Ely Porat and Amir Rothschild. Explicit non-adaptive combinatorial group testing schemes. In *Automata, Languages and Programming*, volume 5125 of *Lecture Notes in Computer Science*, pages 748–759. 2008.

[19] B.S. Tsybakov and V.A. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Prob. Inf. Transmission*, 14(4), April 1978.

[20] Jack K. Wolf. Born again group testing: Multiaccess communications. *IEEE Transaction on Information Theory*, 2, March 1985.

A A Simple Fact about Binomial Coefficients

Proposition A.1. $k \lg \frac{n}{k} \le \lg \binom{n}{k} \le k(\lg \frac{n}{k} + \lg e)$ *for* $k \le \frac{n}{2}$

Proof. By Sterling's approximation,

$$\begin{aligned}
\lg \binom{n}{k} &= (\ln n! - \ln(n-k)! - \ln k!)/\ln 2 \\
&= (n \ln n - n - (n-k)\ln(n-k) + n - k - k \ln k + k)/\ln 2 \\
&= k \lg \frac{n}{k} + (n-k) \lg \frac{n}{n-k} \\
&= k \lg \frac{n}{k} + k \lg(1 + \frac{k}{n-k})^{\frac{n-k}{k}}
\end{aligned}$$

When $k \le \frac{n}{2}$, we have $\frac{n-k}{k} \ge 1$, so $2 \le (1+\frac{k}{n-k})^{\frac{n-k}{k}} < e$, and so $k(\lg \frac{n}{k}+1) \le \lg \binom{n}{k} < k(\lg \frac{n}{k}+\lg e)$. □

B Deterministic CR and MP algorithms in Ternary-feedback Model

A simple counting argument shows that at least $\log_3 \binom{n}{k} \ge k \log_3 \frac{n}{k}$ time slots are required by any deterministic algorithm to solve the membership problem in ternary-feedback model, while there exist algorithms for CR that beat this lower bound (of the membership problem) in the ternary-feedback model. As a simple example, when $k = 1$, only a single time slot is needed for conflict resolution for the ternary-feedback model, while $\Omega(\log n)$ time slots are needed for the membership problem since the single active node's identifier

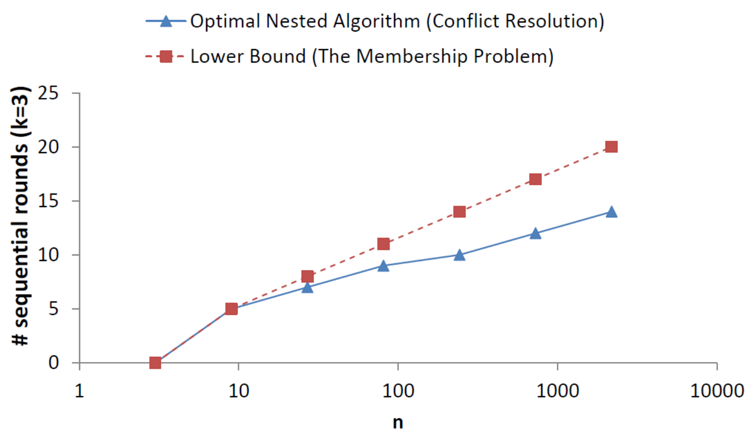

Figure 3: When $k = 3$, the optimal “nested” Conflict Resolution algorithm [7] needs fewer time slots than the lower bound of the Membership Problem in the ternary traditional model, i.e. $\log_3 \binom{n}{k}$.

has to be communicated. For larger values of k, the optimal “nested” algorithm for collision resolution [7] can still beat the lower bound of MP. The following simulation shows a result for $k = 3$. As discussed in the paper, this is different from the Beeping Model, where the two problems are equally hard, with tight lower bounds on both problems.