Optimal Naming in Multi-hop Beeping Networks

Ny Aina Andriambolamalala[1] and Vlady Ravelomanana[2]

[1] IRIF — University Paris Diderot — France. Ny-Aina.Andriambolamalala@irif.fr
[2] IRIF UMR CNRS 8243 — University Paris Diderot — France vlad@irif.fr

Abstract. The naming or initialization problem is one fundamental problem in distributed computing. In this paper, when the nodes do not know neither n nor the number of their neighbors, but all nodes have a unique identifier ID $\in [1, \mathrm{N}]$, we design a deterministic naming and an exact counting algorithm with $O(n + \Delta \log \mathrm{N})$ running times. Our algorithms are optimal in time complexity in view of the $\Omega(\Delta \log \mathrm{N})$ and the $\Omega(n)$ lower bounds given in ([5, 3]). If all nodes are initially indistinguishable and know the diameter D of the network, we adapt our algorithms to have a randomized naming and an exact counting algorithm succeeding with high probability in $O(n + \Delta \log n + D \log \log n)$ time slots.

1 Introduction

In order to compute decentralized tasks in a distributed system, researchers designed various symmetry breaking protocols such as leader election ([4, 15, 13, 21, 20, 10, 11, 16, 23, 22, 14]), maximal Independent Set ([1, 24]) and naming or initialization algorithms ([17, 12, 18, 2, 5]). In this paper, we consider the naming problem on multi-hop beeping networks (the underlying graph of the network is a non oriented connected graph). This consists of assigning to each of the n nodes of the network, a unique label $\ell \in \{1, 2, ...n\}$. In Section 2.2, when no node knows n but all nodes know N and have unique ID $\in [1, N]$ (N is the range of nodes labels, *i.e.*, labels are strings of $O(\log \mathrm{N})$ bits), we design a deterministic naming algorithm terminating in $O(n + \Delta \log \mathrm{N})$ time slots (Δ is the maximum degree of the network). As a consequence, we show how to design a deterministic counting algorithm with $O(n + \Delta \log \mathrm{N})$ time complexity after which all nodes know the exact number of the nodes in the network (Section 2.3). Finally, in Section 3, if all nodes are initially indistinguishable (no node has ID) and no node knows n but all nodes know D (D is the hop diameter of the network), we design randomized naming and counting algorithms terminating in $O(n + \Delta \log n + D \log \log n)$ time slots with high probability [3] (*w.h.p.* for short).

1.1 The model

The beeping model, introduced by Cornejo and Kuhn in 2010 [6], makes little demands on the devices which need only be able to do carrier-sensing, differentiating between silence and the presence of a jamming signal on the network

[3] Event ε_n occurs with high probability if $\mathbb{P}[\varepsilon_n] \geq 1 - \frac{1}{n^c}$ for any constant $c > 0$

(considered as $1-bit$ message or one beep). They note that carrier-sensing can typically be done much more reliably and requires significantly less energy and other resources than message-sending models. In such a model, communications occur in synchronous time slots. In each time slot, a sensor can either beep (transmit $1-bit$ message), listen to the network or remain idle (asleep). Only listening sensors know whether at least one of their neighbors was beeping or all of them remained silent. This model is also called Beep Listen model (BL).

1.2 Related works

In single-hop radio networks (the underlying graph of the network is complete), Hayashi, Nakano and Olariu ([12]) presented a $O(n)$ running time randomized protocol solving the naming problem for the model with collision detection. Later, Bordim, Cui, Hayashi, Nakano and Olariu [2] presented $O(n)$ time complexity algorithm with $O(\log n)$ energy complexity [4]. In [19], Nakano and Olariu designed a naming protocol terminating in $O(n)$ time slots *w.h.p.* with $O(\log \log n)$ energy complexity. Results on beeping model appeared recently when Chlebus, De Marco and Talo [5] presented their naming algorithm terminating in $O(n \log n)$ time slots *w.h.p.* for the BL model and provided a $\Omega(n \log n)$ lower bound on its time complexity. Moreover, Casteigts, Métivier, Robson and Zemmari [3] presented a counting algorithm for the BL model terminating in $O(n \log n)$ time slots *w.h.p.* and an $\Omega(n)$ lower bound. For the multi-hop model, Czumaj and Davies [7] designed a deterministic depth first search(Dfs for short) algorithm initializing the network in $O(n \log \mathrm{N})$ times slots. They start by electing a leader in $O(D \log \mathrm{N})$ time slots, calling the deterministic leader election designed by Förster, Seidel and Wattenhofer [9]. Then, their Dfs algorithm visits the leader first, which takes label 1. To choose which node to visit next, all unlabeled neighbors of the leader have to send their ID bit by bit in order to find which of them has the highest ID. The leader sends a token to such node (let say s) with highest ID and the Dfs algorithm visits s which gets label 2. They recursively applied this algorithm for all newly labeled node by sending a token to its unlabeled neighbor holding the highest ID (the Dfs algorithm visits such a node) and sending back such a token if no unlabeled neighbor remains.

1.3 Our approach

Our main idea is to parallelize the computation choosing which node the Dfs algorithm will visit next. This can be done in $O(\Delta \log \mathrm{N})$ time slots (section 2.1) if the nodes are layered (see below). Having this in mind, we first elect a leader in $O(D + \log \mathrm{N})$ rounds using the algorithm of Dufoulon, Burman and Beauquier [8]. Then, we do a graph layering that consists of putting each node s into a layer $\mathrm{L}(k)$, k being the hop distance between s and the leader. After $O(D)$ time slots of a beep wave [10], the leader belongs to the layer $\mathrm{L}(0)$, all nodes at hop

[4] Energy complexity represents the total number of rounds during which a node is awake either sending a message or listening to the network's channel.

distance 1 from the leader belong to the layer L(1), and recursively, all nodes at hop distance k from the leader belong to the layer L(k). Then, we assign to each node a visit order for the DFS algorithm, in $O(\Delta \log \mathrm{N})$ time slots. Such an ordering consists of assigning an ORDER $\in [0, \Delta]$ to each node such that the DFS algorithm visits the nodes having ORDER 0 first, then ORDER 1 and so on. The goal here is to do these computations in parallel for all nodes so that when the DFS algorithm visits any node, the algorithm knows which node to visit next within $O(1)$ time slots. As a consequence, the precomputation of such visit order leads to a new DFS algorithm terminating in $O(n)$ time slots.

2 N is known and the nodes have unique ID

2.1 Ordering the nodes in advance for the DFS visit

To order each node as described in Section 1.3, we were inspired by the parallelization procedure used in the broadcasting algorithm in [7]. Such a parallelization firstly consists of waking only the nodes in layers $\{\mathrm{L}(0), \mathrm{L}(1)\}$, $\{\mathrm{L}(3), \mathrm{L}(4)\}, ...\{\mathrm{L}(k), \mathrm{L}(k+1)\}$ (such that $k \mod 3 = 0$) to avoid conflicts. The neighbor of any node s in layer $\mathrm{L}(k+1)$ ($s \in \mathrm{L}(k)$) having the highest ID takes order 0, the second highest ID takes order 1 and so on. When the DFS algorithm visits a node v in any layer $\mathrm{L}(l), l \geq 0$, it visits the neighbor of v in layer $\mathrm{L}(l+1)$ having order 0 first. Supposing that all nodes are layered as described in section 1.3 and have already encoded their ID into binary code-word (noted CID), the algorithm is subdivided into at most $O(\Delta)$ seasons $S_0, S_1, ...S_{O(\Delta)}$ (the nodes do not have to know Δ). Each season consists of $\lceil \log \mathrm{N} \rceil + 1$ steps $t_0, t_1, ...t_{\lceil \log \mathrm{N} \rceil}$ and each step t_j is subdivided into four communication time slots $t_{j_0}, t_{j_1} = t_{j_0} + 1, t_{j_2} = t_{j_0} + 2, t_{j_3} = t_{j_0} + 3$.
- at t_{j_0}, the nodes in layer $\mathrm{L}(k), k \geq 0$ beeps in order to signify their neighbors in layer $\mathrm{L}(k+1)$ that they have to take an order.
- unordered nodes in layer $\mathrm{L}(k+1)$ uppon hearing a beep at t_{j_0} beep at t_{j_1} to notify those in layer $\mathrm{L}(k)$ that there remain unordered neighbors.
- at t_{j_2}, the nodes in layer $\mathrm{L}(k+1)$ have to send the j^{th} bit of their CID.
- at t_{j_3}, the nodes in layer $\mathrm{L}(k)$ have to notify the nodes in layer $\mathrm{L}(k+1)$ that at least one node in layer $\mathrm{L}(k+1)$ has CID$[j] = 1$.
Description of the ordering algorithm: At the beginning, for all $i \in \left[0, \lceil \frac{D}{3} \rceil\right]$, all nodes in layers $\mathrm{L}(3i)$ beep in parallel at time slot t_{0_0}. In season S_0, all unordered nodes in layers $\mathrm{L}(3i+1)$ hearing beep at t_{0_0} beeps at t_{0_1} and start sending their CID bit by bit at t_{0_2}. If an unordered node s has CID$[0] = 1$ ($j = 0$), s beeps at t_{0_2}. Each node in layer $\mathrm{L}(3i)$ hearing beep at t_{0_2} beeps at t_{0_3}. Each unordered node in layer $\mathrm{L}(3i+1)$, having CID$[0] = 0$ and hearing beep at t_{0_3} becomes inactive until the end of S_0. The remaining active nodes having CID$[1] = 1$ ($j = 1$) beep at t_{1_2}, each node in layer $\mathrm{L}(3i)$ hearing beep at t_{1_2} beeps at t_{1_3} and so on for all j from 2 to $\lceil \log \mathrm{N} \rceil$. The last remaining active unordered node takes ORDER0, we start season S_1 and so on. After these computations, we have to apply it to layers $\{\mathrm{L}(3i+1), \mathrm{L}(3i+2)\}$ and $\{\mathrm{L}(3i+2), \mathrm{L}(3i+3)\}$ in order to do an ordering for all nodes in all layers.

Algorithm.1 Ordering(N) : Ordering the nodes for Dfs visits

Input : Common value N representing the maximal possible value of ID

Output: Each node s having an order Order

```
s sets ORDER ← −2, status ← NULL, encodes ID into binary code-word CID
for i from 0 to 2 do
    if s in layer L(k) such that k mod 3 = i then
        s sets status ← source, ORDER ← −1, j ← 0, count ← 0, s beeps at t_{0_0}
    end
    if s in layer L(k) such that k mod 3 = i + 1 then
        s sets status ← next, ORDER ← −1, j ← 0, count ← 0
    end
    while (ORDER = −1) ∧ (status ≠ NULL) do
        if (status = next) ∧ (ORDER = −1) then
            s beeps at t_{j_1}
        end
        if (status = next) ∧ (CID[j] = 1) then
            s beeps at t_{j_2}
        end
        if status = source then
            if s doesn't hear beep at t_{j_1} then
                s sets status ← NULL and quits the algorithm
            else
                if s hears beep at t_{j_2} then
                    s beeps at t_{j_3}
                end
            end
        end
        if (status = next) ∧ (CID[j] = 0) ∧ (s hears beep at t_{j_3}) ∧ (j < ⌈log N⌉) then
            s sets status ← inactive
        end
        if j < ⌈log N⌉ then
            j ← j + 1
        else
            if status = next then
                s sets ORDER ← count
            end
            if status = inactive then
                s sets status ← next
            end
            j ← 0, count ← count + 1
        end
    end
end
```

Figure B.1 in Appendix B shows an example of execution of Algorithm.1.

Lemma 1. *Algorithm.1 terminates in $O(\Delta \log \mathrm{N})$ time slots. For any node s in some layer $\mathrm{L}(k)$, let $N_s(k)$ be the set of neighbors of s in layer $\mathrm{L}(k+1)$. After one execution of this algorithm, all nodes $u \in N_s(k)$ have a unique order $\mathrm{ORDER}_u \in \{0, 1, ...|N_s(k)|\}$ such that for any node s, if $u, v \in N_s(k)$, $u \neq v$, $\mathrm{ORDER}_u \neq \mathrm{ORDER}_v$ ($|N_s(k)|$ is the number of nodes in $N_s(k)$).*

Proof. The main idea of Algorithm.1 is that each node has to send the binary encoding of theis ID bit by bit. The nodes which do not have the highest ID are eliminated. By the hypothesis, each ID is unique and after the $\lceil \log \mathrm{N} \rceil$ time slots when each node sent it ID bit by bit, only one node remains. Thus, after each $4 \times \lceil \log \mathrm{N} \rceil$ time slots (4 corresponding to $t_{j_0}, t_{j_1}, t_{j_2}, t_{j_3}$), one node gets an order and quits the algorithm. As a consequence, after $\Delta \times 4 \times \lceil \log \mathrm{N} \rceil$ steps, all nodes in layers $3i+1, i \in [0, \lceil \frac{D}{3} \rceil]$ are ordered. This is done 3 times (line 3 of Algorithm.1) to add order to all the nodes of the network, implying the $3 \times \Delta \times 4 \times \lceil \log \mathrm{N} \rceil = O(\Delta \log \mathrm{N})$ time complexity of the algorithm. □

2.2 Naming algorithm

We start by electing a leader in $O(D + \log \mathrm{N})$ time slots using [8]. Then, we put each node s into a layer $\mathrm{L}(k), 0 \leq k \leq D$ in $O(D)$ rounds (this can be done using for instance the beep waves tool described in [10, paragraph 5.3]). We can now call Algorithm.1 to order all the nodes. With such an ordering, the deterministic naming algorithm visits all nodes one by one starting by the leader by means of a DFS algorithm. Any node s can take one of these 7 *status*:
- ACTIVE if s is the currently visited node.
- Supposing that the ACTIVE node is in any layer $\mathrm{L}(k)$, its neighbors in layer $\mathrm{L}(k+1)$ are called its CHILDREN neighbors.
- ACTIVATED if s was already visited but has remaining unlabeled CHILDREN.
- WAITING if s has never been visited.
- ACTIVELAYER if s is in the same layer as the currently ACTIVE node.
- Supposing that the currently ACTIVE node is in any layer $\mathrm{L}(k)$, s is ACTIVECHILDLAYER if it is in layer $\mathrm{L}(k+1)$.
- DONE if s is already visited and has no remaining unlabeled CHILDREN.
Not to be confused with round's definition in the literature, here, we redefine one round t_j as a set of 14 communication sub-steps $t_{j_0}, ...t_{j_{13}}$, such that $t_{j_i} = t_{j_0} + i$.

- At $t_{j_0}, t_{j_1}, t_{j_2}$, all nodes in any layer $\{\mathrm{L}(0), \mathrm{L}(1)\}, \{\mathrm{L}(3), \mathrm{L}(4)\}, ...\{\mathrm{L}(k), \mathrm{L}(k+1)\}$ such that ($k \mod 3 = 0$) do some computations in parallel to determine if they are ACTIVELAYER or ACTIVECHILDLAYER. At $t_{j_3}, t_{j_4}, t_{j_5}$ (resp. $t_{j_6}, t_{j_7}, t_{j_8}$), we do the same computation for all nodes in any layer $\{\mathrm{L}(1), \mathrm{L}(2)\}$,$\{\mathrm{L}(4), \mathrm{L}(5)\}, ...\{\mathrm{L}(k+1), \mathrm{L}(k+2)\}$ (resp. $\{\mathrm{L}(2), \mathrm{L}(3)\}, \{\mathrm{L}(5), \mathrm{L}(6)\}, ...\{\mathrm{L}(k+2), \mathrm{L}(k+3)\}$).
- At t_{j_9}, the ACTIVE node s (let suppose that s is in any layer $\mathrm{L}(l)$) beeps for asking if there is any unlabeled CHILDREN nodes in layer $\mathrm{L}(l+1)$.

- All unlabeled ACTIVECHILDLAYER nodes in layer $\mathrm{L}(l+1)$ hearing beep at t_{j_9} beep at $t_{j_{10}}$ to notify the ACTIVE node that there remain unlabeled CHILDREN nodes.
- $t_{j_{11}}$ is the feedback step when a CHILDREN node beeps in order to notify the previously ACTIVE node that it gets labeled.
- $t_{j_{12}}$ is the notification step when all nodes hearing beep at $t_{j_{11}}$ beeps in order to notify its CHILDREN that one of their 2-hop neighbors in layer $\mathrm{L}(l+1)$ gets labeled.
- When the ACTIVE node has no more unlabeled CHILDREN, it has to beep in order to notify the precedent ACTIVE node in layer $\mathrm{L}(l-1)$ at $t_{j_{13}}$ which is revisited by the DFS algorithm.

With these 14 sub-steps, we can now explain the execution of our naming algorithm. We first define TEST() protocol for doing all computations during t_{j_0} to t_{j_8} in any round t_j as follows. At t_{j_0}, ACTIVE node in layer $\mathrm{L}(0), \mathrm{L}(3), \ldots \mathrm{L}(k)$(such that $k \mod 3 = 0$), beeps. The WAITING nodes in layers $\mathrm{L}(1), \mathrm{L}(4), \ldots \mathrm{L}(k+1)$ hearing beep at t_{j_0} set *status* $\leftarrow$ ACTIVECHILDLAYER
and beep at t_{j_1}. All WAITING nodes in layer $\mathrm{L}(0), \mathrm{L}(3), \ldots \mathrm{L}(k)$ hearing beep at t_{j_1} set *status* $\leftarrow$ ACTIVELAYER and beep at t_{j_2}. All WAITING nodes in layer $\mathrm{L}(1), \mathrm{L}(4), \ldots \mathrm{L}(k+1)$ hearing beep at t_{j_2} set *status* $\leftarrow$ ACTIVECHILDLAYER and a counter *labeled* $\leftarrow 0$. We do the same computations at $t_{j_3}, t_{j_4}, t_{j_5}$ (resp. $t_{j_6}, t_{j_7}, t_{j_8}$) for layers $\{\mathrm{L}(1), \mathrm{L}(2)\}, \ldots \{\mathrm{L}(4), \mathrm{L}(5)\}, \ldots. \{\mathrm{L}(k+1), \mathrm{L}(k+2)\}$ (resp. $\{\mathrm{L}(2), \mathrm{L}(3)\}, \{\mathrm{L}(5), \mathrm{L}(6)\}, \ldots. \{\mathrm{L}(k+2), \mathrm{L}(k+3)\}$).

Description of the naming algorithm. At the beginning, all nodes set *status* $\leftarrow$ WAITING. The leader gets label $\ell \leftarrow 0$, sets *status* $\leftarrow$ ACTIVE. During t_{0_0} to t_{0_8} $(j = 0)$, all nodes compute their status by evoking *status* $\leftarrow$ TEST(). At t_{0_9}, the ACTIVE node (leader) beeps. All ACTIVECHILDLAYER nodes hearing beep at t_{0_9} set *status* $\leftarrow$ CHILDREN and beep at $t_{0_{10}}$. All new CHILDREN nodes that have no labeled neighbors counter set a counter *labeled* $\leftarrow 0$. The new CHILDREN node having ORDER $=$ *labeled* sets *status* $\leftarrow$ ACTIVE, gets label $\ell \leftarrow 1 = (j+1)$ and beeps at $t_{0_{11}}$. The ACTIVE node hearing beep at $t_{0_{11}}$ sets *status* $\leftarrow$ ACTIVATED and beeps at $t_{0_{12}}$. CHILDREN nodes hearing beep at $t_{1_{12}}$ increment *labeled* $\leftarrow$ *labeled* $+1$. If the ACTIVE node s does not hear a beep at $t_{0_{11}}$, s sets *status* $\leftarrow$ DONE, beeps at $t_{0_{13}}$ and quits the algorithm. All CHILDREN nodes reset *status* $\leftarrow$ WAITING. The ACTIVATED node hearing beep at $t_{0_{13}}$ sets *status* $\leftarrow$ ACTIVE. After that, we do the same computations for the new ACTIVE node as follows. During t_{1_0} to t_{1_8} $(j = 1)$, all nodes compute their status by evoking *status* $\leftarrow$ TEST(). At t_{1_9}, the ACTIVE node (node with label 1) beeps. All ACTIVECHILDLAYER nodes hearing beep at t_{1_9} set *status* $\leftarrow$ CHILDREN and beep at $t_{1_{10}}$. All new CHILDREN nodes that have no labeled neighbors counter set a counter *labeled* $\leftarrow 0$. The new CHILDREN node having ORDER $=$ *labeled* sets *status* $\leftarrow$ ACTIVE, gets label $\ell \leftarrow 2 = (j+1)$ and beeps at $t_{1_{11}}$. the ACTIVE node hearing beep at $t_{1_{11}}$ sets *status* $\leftarrow$ ACTIVATED and beeps at $t_{1_{12}}$. ACTIVELAYER nodes hearing beep at $t_{1_{11}}$ beep at $t_{1_{12}}$. CHILDREN nodes hearing beep at $t_{1_{12}}$ increment *labeled* $\leftarrow$ *labeled* $+1$. ACTIVECHILDLAYER nodes hearing beep at $t_{1_{12}}$ increment *labeled* $\leftarrow$ *labeled* $+1$. If the ACTIVE node s

does not hear a beep at $t_{1_{11}}$, s sets $status \leftarrow$ DONE, beeps at $t_{1_{13}}$ and quits the algorithm. All CHILDREN nodes, ACTIVELAYER nodes and ACTIVECHILDLAYER nodes reset $status \leftarrow$ WAITING. ACTIVATED node hearing beep at $t_{1_{13}}$ sets $status \leftarrow$ ACTIVE. Then we do the same computations for j from 2 to $O(\Delta)$. Figure B.2 in Appendix B shows an example of execution of the 5^{th} round of NAMING() algorithm.

Algorithm.2 TEST() at any node s

Input : Layered nodes having unique ID $\in [1, \mathrm{N}]$
Output: Each node s with $status \in$ {ACTIVE, WAITING, ACTIVELAYER, ACTIVECHILDLAYER}

for *i from* 0 *to* 2 **do**
  if *s is* ACTIVE *and has layer* $\mathrm{L}(k+i)$ *such that* $k \mod 3 = 0$ **then**
    s beeps at $t_{j_{3i}}$
  end
  if *s is* WAITING *and has layer* $\mathrm{L}(k+i+1)$ *such that* $k \mod 3 = 0$ *and hears beep at* $t_{j_{3i}}$ **then**
    s sets $status \leftarrow$ ACTIVECHILDLAYER and beeps at $t_{j_{3i+1}}$
  end
  if *s is* WAITING *and has layer* $\mathrm{L}(k+i)$ *such that* $k \mod 3 = 0$ *and hears beep at* $t_{j_{3i+1}}$ **then**
    s sets $status \leftarrow$ ACTIVELAYER and beeps at $t_{j_{3i+2}}$
  end
  if *s is* WAITING *and has layer* $\mathrm{L}(k+i+1)$ *such that* $k \mod 3 = 0$ *and hears beep at* $t_{j_{3i+2}}$ **then**
    s sets $status \leftarrow$ ACTIVECHILDLAYER, $labeled \leftarrow 0$
  end
end

Lemma 2. *The* WHILE *loop of Algorithm.3 (lines* 9 *to* 41*) terminates in* $O(n)$ *rounds assigning to each node a unique label* $\ell \in [0, 2n]$.

Proof. As the time complexity of DFS algorithm is at most $2n$, the WHILE loop in line 9 of Algorithm.3 terminates after at most $2n$ time slots. We can see in line 20 of Algorithm.3 that the labeling ℓ assigned to each node s corresponds to the time slot when s is visited first by the DFS algorithm. Thus, after the WHILE loop of Algorithm.3, each node has a label $\ell \in [0, 2n]$. □

Lemma 2 show that we don't have a correct labeling ℓ of the network such that $\ell \in \{1, 2, ...n\}$. The following Lemma is important to locally compute such correct labeling without sending any message.

Lemma 3. *For any node s, let $\ell(s)$ be a labeling of s ($\ell(s) \in \{1, 2, ...n\}$, $\ell(s)$ is unique). Let $\ell_A(s)$ (as Assigned label) be the label assigned by the* WHILE *loop of Algorithm.3 to s in any layer* $\mathrm{L}(k)$ *($\ell_A(s) \in [0, 2n]$). For all nodes and all $k \geq 0$,*

$$\ell(s) = \frac{\ell_A(s) + \mathrm{L}(k)}{2} \tag{1}$$

Algorithm.3 NAMING(N) at any node s

Input : A common value N representing the maximal possible value of ID
Output: Each node s having unique label ℓ

s do leader election algorithm as described in [8]
if s *is* LEADER **then**
  s gets layer L(0), $status \leftarrow$ ACTIVE, $\ell \leftarrow 0$, $labeled \leftarrow$ NULL
else
  s gets layer L(k) using the beep waves tool described in [10, paragraph 5.3]), s sets $status$WAITING
end
s sets round counter $j \leftarrow 0$, $num \leftarrow$ ORDERING(N)
while $status \neq$ DONE **do**
  s sets $status \leftarrow$ TEST() during t_{j_0} to t_{j_8}
  if s *is* ACTIVE **then**
    s beeps at t_{j_9}
  end
  if s *is* ACTIVECHILDLAYER *and hears beep at* t_{j_9} **then**
    s sets $status \leftarrow$ CHILDREN and beeps at $t_{j_{10}}$ **if** $labeled =$ NULL **then**
      s sets $labeled \leftarrow 0$
    end
  end
  if s *is* CHILDREN *and labeled* $= num$ **then**
    s sets $status \leftarrow$ ACTIVE, $\ell \leftarrow j+1$ and beeps at $t_{j_{11}}$
  end
  if s *is* ACTIVE **then**
    if s *hears beep at* $t_{j_{11}}$ **then**
      s sets $status \leftarrow$ ACTIVATED and beeps at $t_{j_{12}}$
    else
      s sets $status \leftarrow$ DONE and beeps at $t_{j_{13}}$
    end
  end
  if s *is* ACTIVELAYER *and hears beep at* $t_{j_{11}}$ **then**
    s beeps at $t_{j_{12}}$
  end
  if (s *is* CHILDREN *or* ACTIVECHILDLAYER) *and hears beep at* $t_{j_{12}}$ **then**
    s sets $labeled \leftarrow laleled + 1$
  end
  if s *is* ACTIVATED *and hears beep at* $t_{j_{13}}$ **then**
    s sets $status \leftarrow$ ACTIVE
  end
  if (s *is* CHILDREN *or* ACTIVECHILDLAYER*or* ACTIVELAYER) **then**
    s sets $status \leftarrow$ WAITING
  end
  s sets $j \leftarrow j+1$
end
s sets $\ell \leftarrow \frac{\ell + \mathrm{L}(k)}{2} + 1$

Proof. We first proceed by induction on k. As $\frac{0+0}{2} = 0$, property (1) is always true for $k = 0$. Let now suppose that property (1) is satisfied by every nodes in any layer $\mathrm{L}(k), k > 0$. In layer $\mathrm{L}(k+1)$, all nodes have an ordering $0 \leq \textsc{Order} \leq \Delta$ such that the node having $\textsc{Order} = 0$ is labeled first. We can apply a proof by induction on $\textsc{Order}$ to all nodes in layer $\mathrm{L}(k+1)$. To do so, we start by proving that property (1) is true for all nodes having $\textsc{Order} = 0$.
-*(i)* For any node s having $\textsc{Order} = 0$ in layer $\mathrm{L}(k+1)$, we can considering its parent as root of a network of diameter 2 where all nodes are at distance 1 from the root. As s is visited first by the $\textsc{Dfs}$ algorithm after his parent, we have

$$\ell(s) = \ell(root) + 1 \qquad \text{and} \qquad \ell_A(s) = \ell_A(root) + 1 \tag{2}$$

By our induction hypothesis,

$$\ell(root) = \frac{\ell_A(root) + k}{2}$$

As a consequence,

$$\ell(s) = \frac{\ell_A(root) + k}{2} + 1 = \frac{\ell_A(root) + k + 2}{2} = \frac{\ell_A(root) + 1 + k + 1}{2}$$

Using property (2),

$$\ell(s) = \frac{\ell_A(s) + (k+1)}{2}$$

Thus, property (1) is satisfied for all nodes having $\textsc{Order} = 0$ in $\mathrm{L}(k+1)$.

-*(ii)* Let now suppose that property (1) is satisfied by all nodes having $\textsc{Order} = i, i > 0$ in layer $\mathrm{L}(k+1)$. For any node s, v in layer $\mathrm{L}(k+1)$ having respectively $\textsc{Order} = (i+1)$ and $\textsc{Order} = i$, such that s and v are connected to the same node in layer $\mathrm{L}(k)$, let note $S_T(s)$ (resp. $S_T(v)$) the sub-tree rooted in s (resp. v). $|S_T(s)|$ is the number of nodes in $S_t(s)$. $t_{\textsc{Done}}(s)$ denotes the time slot when a node s is $\textsc{Done}$. We remark that $\ell_A(v)$ can be interpreted as the first time when v is visited. By analyzing the behavior of Algorithm.3, we have

$$\ell_A(s) = t_{\textsc{Done}}(v) + 2 \qquad \text{and} \qquad t_{\textsc{Done}}(v) = \ell_A(v) + 2 \times |S_T(v)|$$

As a consequence,

$$\ell_A(s) = \ell_A(v) + 2 \times |S_T(v)| + 2 \qquad \text{and} \qquad \ell(s) = \ell(v) + |S_T(v)| + 1 \tag{3}$$

By applying our induction hypothesis on $\textsc{Order}$ to (3),

$$\ell(s) = \frac{\ell_A(v) + (k+1)}{2} + |S_T(v)| + 1 = \frac{\ell_A(v) + (k+1) + 2|S_T(v)| + 2}{2}$$

Using (3), we prove that property (1) is true for all nodes in layer $\mathrm{L}(k+1)$. $\square$
We can see an example illustrating Lemma 3 in Figure B.2 (k) of Appendix B.

Theorem 1. *In multi-hop beeping networks of size n where all nodes have unique* ID $\in [0, \mathrm{N}]$ *and know* N*, there is a deterministic naming algorithm, assigning unique label* $\ell \in \{1, 2, ...n\}$ *to all nodes in* $O(n + \Delta \log \mathrm{N})$ *time slots.*

Proof. The time complexity of Algorithm.3 is the sum of
- leader election complexity (line 1 of Algorithm.3) : $O(D + \log \mathrm{N})$
- layering complexity (line 5 of Algorithm.3): $O(D)$
- Ordering(N) complexity (line 8 of Algorithm.3) : $O(\Delta \log \mathrm{N})$ by Lemma 1
- and the While loop in line 9 of Algorithm.3 times Test() complexity ($O(1)$): $O(n)$ by Lemma 2.
By Lemma 3, line 42 of Algorithm.3 assigns to each node a label $\ell \in \{1, 2, ...n\}$ after $O(D + \log \mathrm{N} + D + \Delta \log \mathrm{N} + 2n) = O(n + \Delta \log \mathrm{N})$ time slots without any additional communication. □

2.3 Counting algorithm

Our main idea to design a counting algorithm is to adapt Algorithm.3 such that the last node labeled by Algorithm.3 has to broadcast its label.

Theorem 2. *In multi-hop beeping networks of size n where all nodes have unique* ID $\in [0, \mathrm{N}]$ *and know* N*, there is a deterministic counting algorithm, assigning to all nodes the exact number of participants in* $O(n + \Delta \log \mathrm{N})$ *time slots.*

Proof. we add 2 additional communication steps $t_{j_{14}}, t_{j_{15}}$ to each round t_j of Algorithm.3 in order to have an end notification step for the Naming() algorithm.
- After calling the Ordering(N) algorithm, each node know how many Children nodes it has. Then, we add a Children node counter to each node that decreases every time a Children node gets labeled. When the Leader has only one remaining Children node, it beeps at $t_{j_{14}}$. An unlabeled node s hearing beep at $t_{j_{14}}$ knows that it is last labeled node in layer L(1). When s has only one remaining Children node, it beeps at $t_{j_{14}}$ to notify the last node in layer L(2) and so on.
- An unlabeled node v hearing beep at $t_{j_{14}}$ that no have any Children node know that it is the last labeled node on the network. v starts sending an end notification in $t_{j_{15}}$ and all nodes hearing beep at $t_{j_{15}}$ beep at $t_{(j+1)_{15}}$ and waits for the broadcast from v. v broadcasts ℓ in $O(D + \log n)$ time slots. □

3 n is unknown and the nodes are indistinguishable

If the nodes are indistinguishable and n is unknown, we have to use randomness in order to generate unique ID for breaking symmetry on the network. To do so, the nodes have to know an upper bound of n. In order to find such an upper bound, we use the following results. Let Y be the following discrete probability distribution. Given $\alpha \in]0, 1]$,

$$p_k = \mathbb{P}[Y = k] = e^{-\frac{1+\alpha}{1-\alpha} k} - e^{-\frac{1+\alpha}{1-\alpha}(k+1)}$$

By taking a suitable value of α ($\alpha = \frac{2}{7}$), we have

$$p_k = \mathbb{P}[Y = k] = e^{-\frac{9}{5}k} - e^{-\frac{9}{5}(k+1)} \tag{4}$$

The following Lemma states that the maximal of n copies of random variables distributed as (4) is of the order of $\Theta(\log n)$.

Lemma 4 (Bounds of the maximum). *For $1 \leq j \leq n$, let (Y_j) be n independent copies of a random variable distributed as (4). We have*

$$\mathbb{P}\left[\frac{1}{2}\log n \leq \max_{1\leq j\leq n} Y_j \leq \frac{3}{4}\log n\right] \geq 1 - O\left(\frac{1}{n^{\frac{1}{3}}}\right) \tag{5}$$

Proof.

$$\mathbb{P}\left[\max_{1\leq j\leq n} Y_j \geq \frac{1}{2}\log(n)\right] = 1 - \left(\sum_{j=0}^{\frac{1}{2}\log(n)} e^{-\frac{9}{5}j} - e^{-\frac{9}{5}(j+1)}\right)^n = 1 - \left(1 - e^{-\frac{9}{5}(\frac{1}{2}\log(n)+1)}\right)^n.$$

Since $(1-x) \leq e^{-x}$, after a bit algebra, we have

$$\mathbb{P}\left[\max_{1\leq j\leq n} Y_j \geq \frac{1}{2}\log(n)\right] \geq 1 - (e^{-ne^{-\frac{9}{5}(\frac{1}{2}\log(n)+1)}})^n \geq 1 - O\left(\frac{1}{e^{n^{\frac{1}{10}}}}\right)$$

In the same way, since $(1-x) \geq e^{-x-x^2}$, the probability that the maximum is at most $\frac{3}{4}\log(n)$ is

$$\left(\sum_{\delta=0}^{\frac{3}{4}\log(n)} e^{-\frac{9}{5}\delta} - e^{-\frac{9}{5}(j+1)}\right)^n = \left(1 - e^{-\frac{9}{5}(\frac{3}{4}\log(n)+1)}\right)^n \geq e^{-n^{-\frac{7}{20}} - n^{-\frac{34}{20}}} \geq 1 - O\left(\frac{1}{n^{\frac{1}{3}}}\right).$$

$\square$

We can now find an upper bound u of n in each node of the network as follows.

3.1 Finding u such that $n^2 \leq u \leq n^3$

Each node s generates one local random variable Y_s distributed as (4). Then, all nodes start finding an upper bound of the maximum of all generated *r.v.* as follows. At the beginning, all nodes having $Y_s \in [0, 2]$ beeps. The other nodes have to listen to the network during D time slots and to beep immediately after hearing a beep. Thus, after D time slots, all nodes know if at least one of them have $Y_s \in [0, 2]$. We do the same computation for $[0, 4], [0, 8]$... until reaching the interval in which no node has Y_s. Then the nodes do a dichotomy on this last interval to find $\max_{1\leq s\leq n} Y_s$. In what follows, TestBeep(a, b) (see Appendix A) is a local operation called at some time slot t_k. It works during D time slots $t_k, t_{k+1}, ...t_{k+D-1}$. During such time slots, a node s beeps at the beginning (at t_k) if its random variable Y_s verifies $a \leq Y_s \leq b$, otherwise s listens during

D time slots. Each node hearing beep at any round immediately beeps after. After any D time slots, TestBeep(a, b) outputs Beep if and only if at least one node of the network beeps during these rounds (that is at least one node v has $a \leq Y_v \leq b$). Observe that during such a rounds, all beeping nodes know they just beeped.

Lemma 5. *Algorithm 5(see Appendix A) terminates in $O(D \log\log n)$ w.h.p.*

Proof. By Lemma 4, the maximum of all generated *r.v.* by all the nodes is in $[\frac{1}{2}\log n, \frac{3}{4}\log n]$ *w.h.p.* As a consequence, finding its upper bound as in lines 2 to 4 of Algorithm 5 ends after $O(\log\log n)$ loops. In the same way, finding the exact value of such a maximum by mean of a dichotomy ends after $O(\log\log n)$ loops (lines 7 13 of Algorithm 5). As in each loop, all nodes call the TestBeep(a, b) algorithm, having $O(D)$ time complexity, Algorithm 5 terminates *w.h.p.* after $O(D \log\log n)$ rounds. Thus, by Lemma 4, the last line of Algorithm 5 provides $n^2 \leq u \leq n^3$ *w.h.p.*. □

3.2 Naming and counting algorithms

Theorem 3. *In multi-hop beeping networks of size n where all nodes are initially indistinguishable, there is a randomized naming algorithm, assigning unique label $\ell \in \{1, 2, ...n\}$ to all nodes in $O(n + \Delta \log \mathrm{N} + D \log\log n)$ time slots w.h.p.*

Proof. After calling the previous protocols, all nodes can generate unique ID s *w.h.p.* and can know N, the maximal range of all ID s. Thus, all the nodes can execute the Naming(N) algorithm. As a consequence, the time complexity of sych a randomized naming algorithm is $O(n + \Delta \log \mathrm{N} + D \log\log n)$ □

Theorem 4. *In multi-hop beeping networks of size n where all nodes nodes are initially indistinguishable, there is a randomized counting algorithm, assigning to all nodes the exact number of participants in $O(n + \Delta \log \mathrm{N} + D \log\log n)$ time slots w.h.p..*

Proof. As the deterministic version, our randomized naming algorithm can be adapted to have a randomized counting algorithm with $O(n + \Delta \log \mathrm{N} + D \log\log n)$ time complexity.

4 Conclusion

In this paper, we presented deterministic naming algorithm on multi-hop beeping networks. When no node knows the size n of the network, and all nodes have unique ID $\in [1, \mathrm{N}]$, our protocol terminates in optimal $O(n + \Delta \log \mathrm{N})$ rounds assigning unique label $\ell \in \{1, 2, ...n\}$ to each node. Our algorithm can be adapted for the counting problem, returning the exact number of nodes in $O(n + \Delta \log \mathrm{N})$ rounds. All these algorithms are optimal in time complexity in view of the $\Omega(\Delta \log \mathrm{N})$ lower bound and the $\Omega(n)$ lower bound given in ([5, 3]). In the case when the nodes are indistinguishable, we design randomized naming and counting algorithms terminating in $O(n + \Delta \log \mathrm{N} + D \log\log n)$ time slots *w.h.p.*.

References

1. Afek, Y., Alon, N., Bar-Joseph, Z., Cornejo, A., Haeupler, B., Kuhn, F.: Beeping a maximal independent set. Distributed computing **26**(4), 195–208 (2013)
2. Bordim, J.L., Cui, J., Hayashi, T., Nakano, K., Olariu, S.: Energy-efficient initialization protocols for ad-hoc radio networks. In: International Symposium on Algorithms and Computation. pp. 215–224. Springer (1999)
3. Casteigts, A., Métivier, Y., Robson, J.M., Zemmari, A.: Counting in one-hop beeping networks. to appear Theoretical Computer Science (2016)
4. Chang, Y.J., Kopelowitz, T., Pettie, S., Wang, R., Zhan, W.: Exponential separations in the energy complexity of leader election. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. pp. 771–783. ACM (2017)
5. Chlebus, B.S., De Marco, G., Talo, M.: Naming a channel with beeps. Fundamenta Informaticae **153**(3), 199–219 (2017)
6. Cornejo, A., Kuhn, F.: Deploying wireless networks with beeps. In: International Symposium on Distributed Computing. pp. 148–162 (2010)
7. Czumaj, A., Davies, P.: Communicating with beeps. Journal of Parallel and Distributed Computing (2019)
8. Dufoulon, F., Burman, J., Beauquier, J.: Beeping a deterministic time-optimal leader election. In: 32nd International Symposium on Distributed Computing (DISC 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2018)
9. Förster, K.T., Seidel, J., Wattenhofer, R.: Deterministic leader election in multi-hop beeping networks. In: International Symposium on Distributed Computing. pp. 212–226. Springer (2014)
10. Ghaffari, M., Haeupler, B.: Near optimal leader election in multi-hop radio networks. In: Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms. pp. 748–766 (2013)
11. Ghaffari, M., Lynch, N., Sastry, S.: Leader election using loneliness detection. Distributed Computing **25**(6), 427–450 (2012)
12. Hayashi, T., Nakano, K., Olariu, S.: Randomized initialization protocols for packet radio networks. In: ipps. p. 544. IEEE (1999)
13. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Efficient algorithms for leader election in radio networks. In: Proceedings of the twenty-first annual symposium on Principles of distributed computing. pp. 51–57. ACM (2002)
14. Kardas, M., Klonowski, M., Pajak, D.: Energy-efficient leader election protocols for single-hop radio networks. In: Parallel Processing (ICPP), 2013 42nd International Conference on. pp. 399–408. IEEE (2013)
15. Kutten, S., Pandurangan, G., Peleg, D., Robinson, P., Trehan, A.: Sublinear bounds for randomized leader election. In: International Conference on Distributed Computing and Networking. pp. 348–362. Springer (2013)
16. Lavault, C., Marckert, J.F., Ravelomanana, V.: Quasi-optimal energy-efficient leader election algorithms in radio networks. Journal of Information and Computation **205**(5), pages–679 (2007)
17. Nakano, K.: Optimal initializing algorithms for a reconfigurable mesh. Journal of Parallel and Distributed Computing **24**(2), 218–223 (1995)
18. Nakano, K., Olariu, S.: Energy-efficient initialization protocols for radio networks with no collision detection. In: International Conference on Parallel Processing, 2000. pp. 263–270 (2000)

19. Nakano, K., Olariu, S.: Energy-efficient initialization protocols for single-hop radio networks with no collision detection. IEEE Transactions on Parallel and Distributed Systems **11**(8), 851–863 (2000)
20. Nakano, K., Olariu, S.: Randomized leader election protocols in radio networks with no collision detection. In: International Symposium on Algorithms and Computation. pp. 362–373 (2000)
21. Nakano, K., Olariu, S.: A survey on leader election protocols for radio networks. In: International Symposium on Parallel Architectures, Algorithms and Networks. I-SPAN'02. pp. 71–76. IEEE (2002)
22. Nakano, K., Olariu, S.: Uniform leader election protocols for radio networks. IEEE Trans. on Parallel Distrib. Syst. **13**(5), 516 – 526 (2002)
23. Ramanathan, M.K., Ferreira, R.A., Jagannathan, S., Grama, A., Szpankowski, W.: Randomized leader election. Distributed Computing **19**(5-6), 403–418 (2007)
24. Scott, A., Jeavons, P., Xu, L.: Feedback from nature: an optimal distributed algorithm for maximal independent set selection. In: Proceedings of the 2013 ACM symposium on Principles of distributed computing. pp. 147–156. ACM (2013)

Appendix

A Randomized algorithms implementations

Algorithm 4. TestBeep(a, b).

Input : Each node s with a local random variable Y_s, 2 values a and b
Output: Beep or Null

if $Y_s \in [a, b]$ **then**
  Beep at t_0
  return Beep
else
  for j *from* 1 *to* D **do**
    if *heard* Beep *at* t_{j-1} **then**
      Beep at t_j
      return Beep
    else
    end
  end
end
return Null

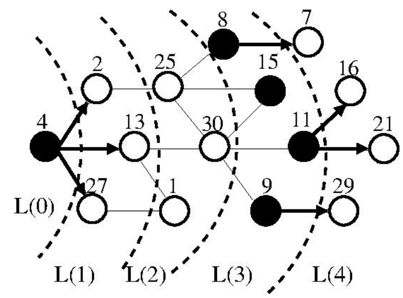
Algorithm 5. Finding u.

Input : A common constants $\alpha = \frac{2}{7}$, the diameter D of the network
Output: Each node s knowing a common value $u \in [n^2, n^3]$.

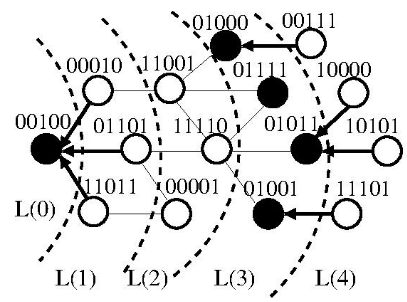
Each node s generates one $r.v.$ Y_s as defined by (4) and sets status $\leftarrow$ Active.
All nodes set $k \leftarrow 0$ and **do**
  s sets $k \leftarrow k + 1$, Sup $\leftarrow 2^k$
while Test(0, Sup) $\neq$ Null
Each node s stores the last value of k
All nodes set Inf $\leftarrow 0$ and Sup $\leftarrow 2^k$.
while Inf $\neq$ Sup **do**
  if Test($\lceil(\text{Inf} + \text{Sup})/2\rceil$, Sup) = Null **then**
    Sup $\leftarrow \lceil \frac{\text{Inf}+\text{Sup}}{2} \rceil - 1$
  else
    Inf $\leftarrow \lceil \frac{\text{Inf}+\text{Sup}}{2} \rceil$
  end
end
All nodes set $u \leftarrow 2^{4\times \text{Sup}}$

B Figures

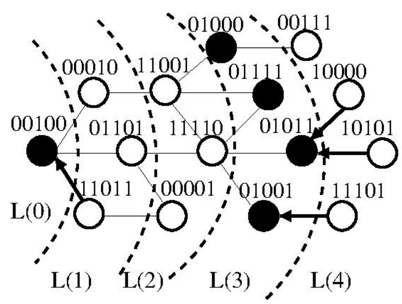
B.1 Example of execution of Ordering(N) algorithm



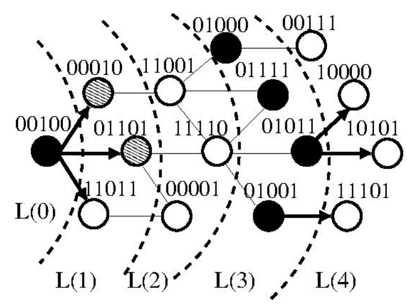
t_{0_0}: Nodes in L(0), L(3) (black nodes) beep



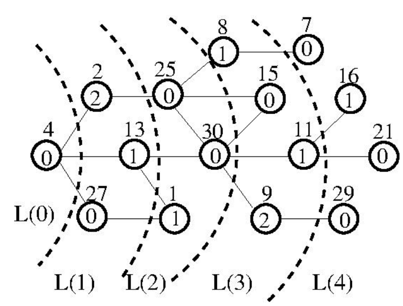
t_{0_1}: Unordered nodes in in L(1), L(4) beep



t_{0_2}: Unordered nodes inL(1), L(4) with CID[0] = 1 beep

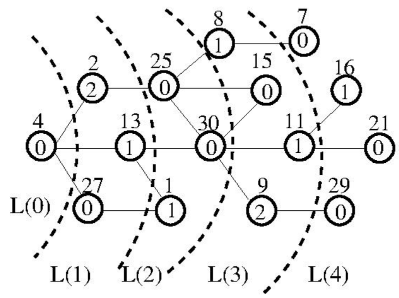


t_{0_3}: Nodes in L(0), L(3) hearing beep at t_{0_2} beep and nodes in L(1), L(4) with CID[0] = 0 (striped nodes) are eliminated

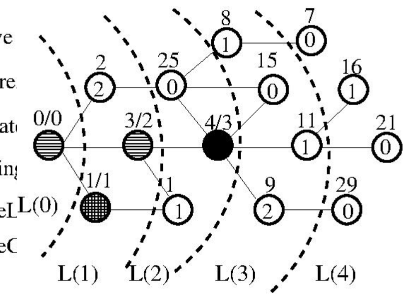


At the end of Algorithm.1, all nodes have an order

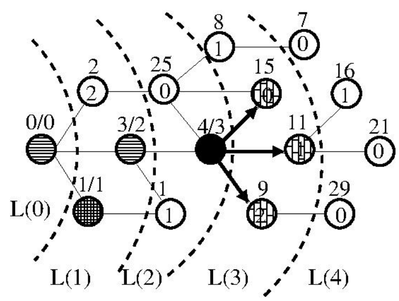
B.2 Example of execution of the 5^{th} round of Naming() algorithm. We only show steps $t_{5_6}, t_{5_7}, t_{5_8}$ for Test() algorithm because the current Active node is in layer L(2). In the notation A/B, A corresponds to label assigned by the While loop and B represents the label $\ell \in \{1, 2...n\}$.

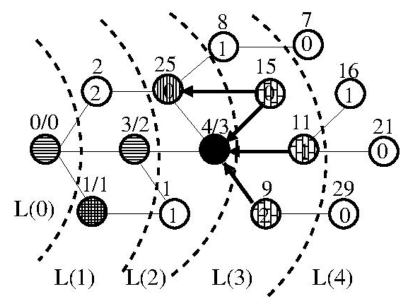


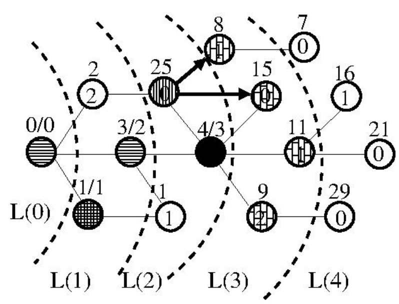
(a) The network after Ordering(N) algorithm

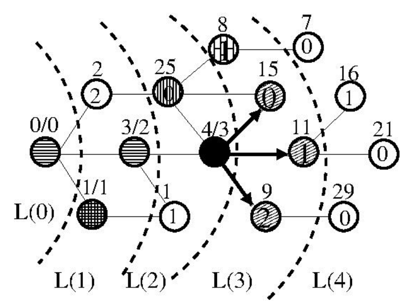


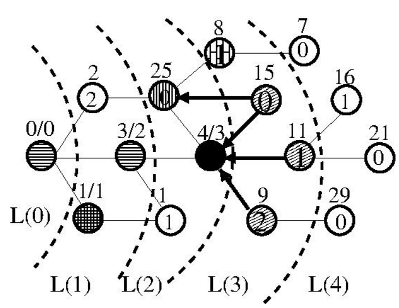
(b) After round t_4 of Naming(N) algorithm

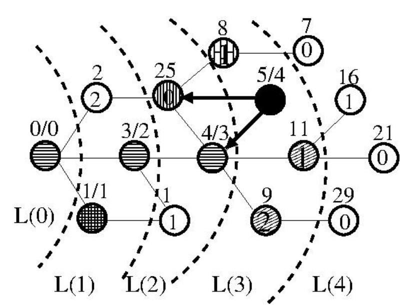

(c) Step t_{5_6} of Test()


(d) Step t_{5_7} of Test()

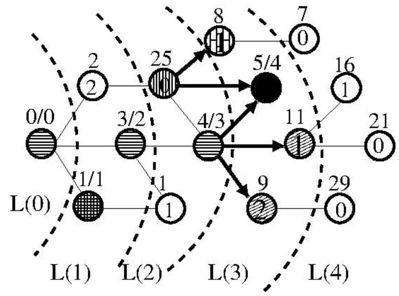

(e) Step t_{5_8} of Test()


(f) Step t_{5_9} just after Test()

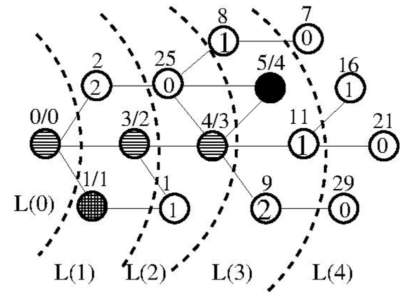

(g) Step $t_{5_{10}}$



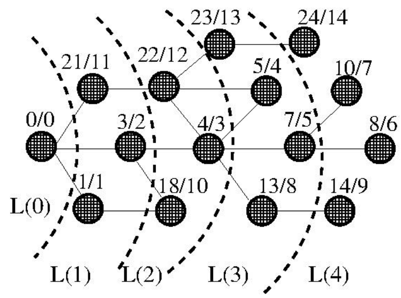
(h) Step $t_{5_{11}}$



(i) Step $t_{5_{12}}$



(j) Step $t_{5_{13}}$



(k) Network after Naming(N)