An Energy-Efficient Initialization Algorithm for Random Radio Networks

Binh T. Doan, Christian Lavault, Stephan Olariu, Vlady Ravalomanana

An Energy-Efficient Initialization Algorithm for Random Radio Networks

Binh Thanh DOAN, Christian LAVAULT, Stephan OLARIU, Vlady RAVELOMANANA

***Abstract*— A radio network is a distributed system consisting of a large number of tiny sensors with low-power transceivers and no central controller. One of the most important problems in such networks is to minimize the energy consumption, and maximize the network lifetime. In the initialization problem (also known as naming) each of the n indistinguishable (anonymous) nodes in a given network is assigned a unique identifier, ranging from 1 to n. We consider a network where n nodes (processors) are randomly deployed in a square (resp. a cube) X. The network is assumed to be synchronous and the time to be slotted. Two nodes can communicate only if they are at a distance of at most r from each other (r is the transmitting/receiving range). Moreover, if two or more neighbors of a processor u are transmitting concurrently at the same time slot, u cannot receive either of their messages (collision). We suppose also n and $|X|$ represent the only topological knowledge in each node. To solve the initialization problem, we propose an energy-efficient randomized algorithm running in at most $\mathcal{O}\left(n^{3/4}\,\log(n)^{1/4}\right)$ time slots, with no station being awake for more than $\mathcal{O}\left(n^{1/4}\,\log(n)^{3/4}\right)$ time slots.**

***Index Terms*— Multihop networks; self-configuration in ad hoc networks; randomized distributed protocols; initialization; naming; energy efficient algorithms.**

I. INTRODUCTION

Distributed multihop wireless networks, such as ad hoc networks sensor networks or radio networks, are gaining in importance as subject of research [20]. Here, a network is a collection of transmitter-receiver devices, referred to as *nodes* (*stations* or *processors*). Wireless multihop networks are formed by a group of nodes that can communicate with each other over a wireless channel. Nodes or processors come without ready-made links and without centralized controller. The network formed by these processors can be modeled by its *reachability graph* in which the existence of a directed arc $u \to v$ means that v can be reached from u. If the power of each transmitter/receiver is the same, the underlying reachability graph is symmetric. As opposed to traditional networks, wireless networks are often composed of nodes whose number can be several orders of magnitude higher than the nodes in conventional networks [1]. Sensor nodes are often deployed inside a medium. Therefore, the positions of these nodes need not be engineered or pre-determined. This allows a random and rapid deployment in inaccessible terrains and is well suited to the specific needs for disaster-relief, law enforcement, collaborative computing and other special purpose applications.

As customary [2]–[6], [9], [14], [15] the time is assumed to be slotted and nodes can send messages in synchronous *rounds* or *time slots*. In each round, every node can act either as a *transmitter* or as *receiver*. During a round a station might be either *awake* or *asleep* but a sleeping station is totally unreachable. As in [3], [4], [24], we also assume that the amount of information that can be sent by a node at each time slot is *unlimited*. A node u acting as receiver in a given round gets a message if, and only if, only one of its neighbors is transmitting in the same round. If at least two neighbors of u are transmitting simultaneously, u receives nothing. In other words, the networks is considered not to be able to distinguish between an absence of message and collision(s) (or conflicts). This assumption is motivated by the fact that in many real-life situations, the (tiny) devices used do not always have the capacities to carry out collision detection. Moreover, even if such a detection mechanism were available, it might prove irrelevant, especially in the presence of some noisy channels. It is thus highly desirable to design protocols that are working independently of any collision detection capability.

We consider that a set of n nodes are initially *homogeneously scattered* in a square X of size $|X|$ (or in a cube X of volume $|X|$). As in several applications, the entities can move within the network; so the topology is unstable. For this reason, we must refrain from assuming too much about the knowledge

Binh Thanh Doan is with the Institut de la Francophonie pour l'Informatique (IFI), Hanoi, Vietnam. E-mail : dtbinh@ifi.edu.vn.

Stephan Olariu is with the Department of Computer Science, Old Dominion University, Norfolk, VA 23529. E-mail : olariu@cs.odu.edu.

Vlady Ravelomanana and Christian Lavault are with the LIPN – UMR 7030 (CNRS), Institut Galilée Université de Paris 13, France. E-mail : {vlad,lavault}@lipn.univ-paris13.fr.

of the network topology in the design of protocols. In this work, the nodes are assumed to have little initial structural information on the network (such as topological knowledge for example); more precisely, every node knows only the number n of participating stations and the measure $|X|$ of the surface X where they are randomly deployed.

Methods to achieve *self-configuration* and/or *self-organization* of networking devices appear to be amongst the most important challenges in wireless computing [1]. The initialization (or naming) task is part of these methods: before networking, each node must have a *unique identifier* (referred to as *ID* or *address*). A mechanism that allows the network to create a unique address (ID) automatically for each of its participating nodes is called the *address autoconfiguration* protocol. In this work, our node are initially *indistinguishable*. This assumption arises naturally since it may be either difficult or impossible to get interface serial numbers while on missions (see also [9], [14], [15]). Thus, the IDs self-configuration protocols do not have to rely on the existence of serial numbers.

Previous works.

The problem we address here is to design an *energy-efficient distributed protocol* for the initialization problem (also known as *naming* problem). As far as we know, the initialization problem for radio networks was first handled in the seminal papers of Hayashi, Nakano and Olariu [9], [14] for the case when the underlying reachability graph is a complete one. Then, Nakano and Olariu [15] designed an energy-efficient protocol for this problem[1]. In the case of randomly scattered nodes, Ravelomanana [22] presented the first two sublinear randomized initialization algorithms, running in $\mathcal{O}\left(n^{1/2}\log n^{1/2}\right)$ and $\mathcal{O}\left(n^{1/3}\log n^{2/3}\right)$ rounds (resp.), whenever the support X is a square or a cube (resp.). The algorithms in [22] are not energy-efficient, since all stations remain awake during the whole execution of both protocols.

Our results.

Given a square X of size $|X|$, $n = \mathcal{O}(|X|)$ nodes are randomly deployed in X with transmission radius $r = \sqrt{\frac{(1+\ell)\log(n)|X|}{\pi\, n}}$ ($\ell > 0$).

We present a randomized algorithm running in $\mathcal{O}\left(n^{3/4}\,\log(n)^{1/4}\right)$ rounds, with no station being awake for more than $\mathcal{O}\left(n^{1/4}\,\log(n)^{3/4}\right)$ time-slots. It is shown that our sublinear and energy-efficient initialization protocol is at most $\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ far from optimality, with respect to the number of rounds required. In fact, the running time is $\mathcal{O}(D\log n) = \mathcal{O}(D\Delta)$, where Δ is the maximum degree of the underlying network and D denotes its eccentricity (hop-diameter). Indeed it was shown in [11] that, in the same setting, the easiest broadcasting problem requires $\Omega\left(D\log\log n\right)$ rounds.

Outline of the paper.

The remainder of this extended abstract is organized as follows. Section 2 is devoted to examine the principal issue for solving the problem; and the main steps PREPARATION, CLUSTERING, LOCAL INITIALIZATION, GLOBAL INITIALIZATION of our result are also presented. The ideas that lay behind these steps can be briefly sketched as follows.

In order to schedule all communications, we color the network stations in such a way that any pair (u, v) of nodes at distance ≤ 2 are assigned two distinct colors. This coloring algorithm suggests a natural scheduling of all the communications in our protocols. This specific algorithm and some others are termed as the PREPARATION protocols.

Next, the divide-and-conquer principle is applied. We cluster the graph (CLUSTERING), with a specific node called the *cluster head* in each cluster. Every cluster is then locally initialized (LOCAL INITIALIZATION). Finally, the global initialization step (GLOBAL INITIALIZATION) is carried out over the graph of clusters. All communications are realized via the specific paths constructed between neighboring clusters (this by swapping from one to one). A gossiping algorithm between all cluster heads, just followed by a ranking algorithm complete this last step.

All details regarding the design and analysis of protocols are provided in Section 3. Section 4 proposes some final concluding remarks as well as open problems.

II. MAIN STEPS

A. Fundamental characteristics of the network

The n nodes are deployed randomly in a given square X with size $|X|$ and $n = \mathcal{O}(|X|)$. Each node has a transmission radius $r = \sqrt{\frac{(1+\ell)\log(n)|X|}{\pi\, n}}$.

Under this setting, we know that with high probability[2], the underlying reachability graph satisfies the following main characteristics.

[1]The energy saving efficiency of the algorithms is measured as the number of rounds required to achieve the designated tasks.

[2]Throughout the paper, an event $\mathcal{E}_n$ is said to occur *asymptotically almost surely* if, and only if, $\mathbb{P}(\mathcal{E}_n) \to 1$ as $n \to \infty$. We also say that $\mathcal{E}_n$ occurs *with high probability* (*w.h.p.* for short).

• There exist two constants c_ℓ and C_ℓ such that the degree d_v of any node v meets the condition

$$c_\ell \log n \leq d_v \leq C_\ell \log n.$$

• The graph is $(c_\ell \log n)$-connected.
• If D denotes the hop-diameter (or diameter) of the network, $D = \Theta(\log n)$.
The reader may refer to [7], [8], [18], [19], [21] and references therein concerning the above characteristics.

B. Preparation

Since all nodes in the radio network are indistinguishable, our first task is to assign temporary distinct IDs (TMPIDs) to all of them. With this end, and since n is known to each station, each of them is allowed to choose randomly, independently and uniformly a temporary ID (TMPID) from a (large enough) set, say $[1, n^3]$. It can be shown that in such a process, *w.h.p.* no nodes can draw the same TMPID (see also [22]).
On the other hand, collisions can occur in radio networks when two nodes are trying to transmit to a common neighbor at one same time slot. By applying the procedure ASSIGNCOLOR given in [22], we assign colors to the nodes of the network; ASSIGNCOLOR is a 2-hop coloration algorithm. After its execution, any two nodes at a distance of at most two hops from each other are assigned two distinct colors (or codes).
Furthermore, this algorithm induces a natural collision-free scheduling: each node u with color $c(u)$ is allowed to send a message *iff* TIME MOD $c(u) \equiv 0$. (The protocol TIME gives any such node the knowledge of a current global time).

C. Clustering

The aim of this step is to design a randomized algorithm that partitions the set of nodes into disjoint groups. The hop-diameter of each group ranges between k and $2k$, where k is a parameter that will be fixed later in the analysis of the algorithm. In each cluster, there is a specific node called the cluster head (CH for short). The principle of the clustering protocol is simple and intuitive.
At first, each node becomes a candidate cluster head with a certain probability p. If two or more candidate cluster heads are too close from each other (viz. they are within less than k-hops distance), all of them must be eliminated but one, which is considered the true cluster head. This can be done by choosing the candidate with the biggest TMPID amongst all others, and the eliminated candidates become normal stations.
At the end of the algorithm, we have to collect all the *orphan* nodes, that is all the nodes which are not in the k-hop neighborhood of the newly appointed CH. The orphans choose the nearest CH among all the possible cluster heads in their respective $2k$-hop neighborhoods. During the clustering protocol, every communication is mainly performed by using the 2-hop coloration algorithm mentioned above.
This partitioning process is a key ingredient of our initialization algorithm. After the execution of the clustering protocol, each cluster can be initialized locally.

D. Local initialization

In order to initialize each cluster locally, the protocol GOSSIP is used. The idea is very similar to those in [22]. The local initialization protocol is executed distributively by all the stations in all the clusters. Every node in the network transmits its TMPID to all other stations. The gossiping protocol uses the collision-free scheduler that the coloration algorithm provides, and when a node receives a message msg, it appends its TMPID to msg and transmits this new message to all its neighbors. Since the coloration algorithm uses $\mathcal{O}(\log n)$ colors, after $\mathcal{O}(k \log n)$ rounds, all stations know the TMPIDs of all other stations in their cluster. Finally, the rank of the TMPID of a node just becomes its local ID (denoted LOCALID).
Upon termination of the local initialization step, each node owns two "IDs": its temporary ID (TMPID) and its local ID (LOCALID), according to the cluster where it belongs.

E. Paths between the clusters

In the next step, the paths of communication between each cluster must be constructed. The idea is as follows. During such communications, all nodes are intentionally *asleep* to save their batteries, except the nodes on these paths. To avoid "energy holes" (on the most crowded paths), we have to find out as many disjoint paths as possible and swap over from one path to another.

F. Global initialization

The global initialization step is performed by means of a gossiping algorithm between all cluster heads, just followed by a ranking algorithm. All communications are carried out via the disjoint paths between clusters, and by skipping from one path to the other.

The following figures 1 and 2 describe and briefly summarize the main steps of the initialization protocol.

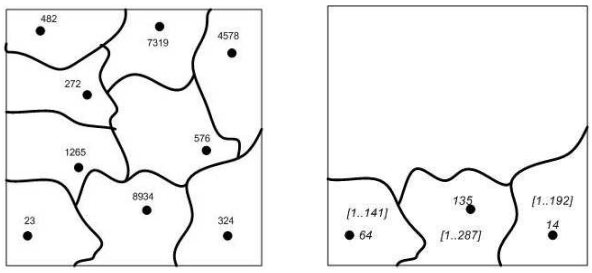

Fig. 1. Division of the graph into disjoint clusters and local initialization of each cluster. In the figure on the right above, 3 clusters are initialized with integers ranging from 1 to 141, from 1 to 287, and from 1 to 192, respectively.

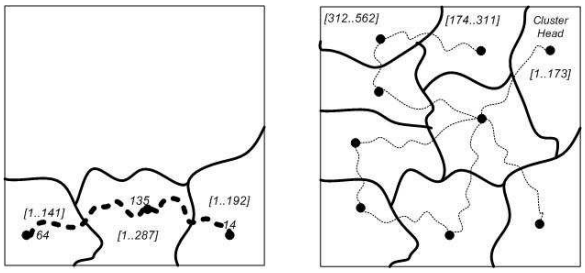

Fig. 2. After the paths construction between clusters (dashed lines), the global initialization is just executed by means of the gossiping algorithm performed via these paths.

III. Detailed Algorithms and Analysis

A. Coloring, broadcasting and gossiping

In this paragraph, we are concerned with 3 basic protocols which are frequently used and discussed throughout the remainder of the paper. First, the protocol AssignColor is executed (see the design in [22, Section VI]). AssignColor requires $\mathcal{O}\left(\log n^4\right)$ time slots (rounds) and it uses $\mathcal{O}\left(\log n\right)$ colors. After the execution of AssignColor, any two nodes within 2 hops-distance received two distinct codes (colors) with high probability. Once well-colored, the network is collision-free and we are now ready to design the protocol BROADCAST. (The pseudo-code is in Fig. 3.) It is easily seen that such a protocol requires $\mathcal{O}(k \log n)$ rounds, under the condition that the randomized coloring algorithm succeeds with probability 1 (i.e, it errs with probability 0).

```
1  Procedure BROADCAST(msg :: message,
                       k :: integer)
2  Begin
3   Repeat (100 × k × log n) times
4    For a node u of color c(u), upon receiving
     a message of the form ⟨msg, k⟩ Do
5      If (TIME mod c(u)) ≡ 0 and k > 0 Then
6        BROADCAST(msg, k − 1)
7   End Repeat
8  End.
```

Fig. 3. The BROADCASTING algorithm.

One can design a gossip algorithm based upon BROADCAST. In the gossiping problem, the task is to spead out the information contained in each node to all the others. Such a protocol can be derived from the broadcasting one by changing a few lines, as described in Fig. 4.

```
1  Procedure GOSSIP(k :: integer)
2  Begin
3   Repeat (100 × k × log n) times
4    For each node u with initial message
     msg(u) and color c(u) upon receiving
     any message of the form ⟨msg, k⟩ Do
5      If (TIME mod c(u)) ≡ 0 and k > 0 Then
6        msg := append(msg, msg(u)) ;
7        TRANSMIT(msg) ;
8        GOSSIP(k − 1) ;
9   End Repeat
10  End.
```

Fig. 4. The GOSSIPING algorithm.

Since there are $\mathcal{O}(\log n)$ colors and k steps, the execution time of GOSSIP(k) is the same as BROADCAST(k): $\mathcal{O}(k \log n)$.

B. Random clustering

In order to apply a divide-and-conquer algorithm, we design the protocol CLUSTERING, which works as follows.

At first, each station chooses to be a candidate cluster head (CH) with a certain probability p (which is specified later in the analysis). The protocol meets the following specifications:

(i) each cluster has a CH;
(ii) each node knows its CH, which is at most within $2k$ hops-distance;
(iii) any two CHs are at a distance of at least $k+1$ hops from each other.

Therefore, there exist randomly chosen candidates in the support area X. In order to satisfy specification (iii) given above, a few candidates which are too close from each other must be eliminated.

By using a broadcasting protocol at a distance k (which can be done with BROADCAST), each candidate CH can detect whether there exist some other candidates in its k-hop neighborhood. The candidate with the biggest TMPID becomes a true CH and all others are eliminated.

Finally, the orphans (stations without CH) are collected as follows (COLLECT). Every CH executes a protocol, with a specific message that enables each orphan to choose the nearest CH in its $2k$-neighborhood.

```
1  Procedure COLLECT(j :: integer)
2  Begin
3    For each node u Do
4    If u is a cluster head Then
5     Repeat (100 × j × log n) times
6       If (TIME mod c(u)) ≡ 0 Then
7         TRANSMIT(TMPID,1)
8     End Repeat
9    Else
10      CLUSTER(u) := NIL, distance := ∞;
11     Upon receiving a message of the form
           ⟨TMPID, radius⟩
12      If distance > radius Then
13     CLUSTER(u) := TMPID, distance := radius;
14       Repeat (100 × j × log n) times
15         If (TIME mod c(u)) ≡ 0 Then
16         TRANSMIT(CLUSTER(u),distance+1);
17       End Repeat
18      End If
19    End Else
20 End.
```

The protocol CLUSTERING (pseudo-code) is then as follows.

```
1  Procedure CLUSTERING(k :: integer, p ::
   float)
2  Begin
3  Step 1: Each station chooses to be a
   CANDIDATE cluster head with probability p;
4  Step 2: For each CANDIDATE station run
   BROADCAST(TMPID,k);
5  Step 3: Upon receiving a broadcasting mes-
   sage, eliminate the candidates which TMPID is
   smaller than the ID(s) of some other(s) candi-
   date(s).
6  Step 4: The remaining candidates are now
   cluster heads and broadcast their TMPID by
   means of COLLECT(TMPID, 2k), to inform the
   stations at 2k-hop distance of their presence and
   status.
7  Step 5: For each node u, CLUSTER(u) is set
   to the nearest cluster head among the nodes that
   invocated the protocol COLLECT.
8  End.
```

From Section B, we derive the following result.

Theorem 1: CLUSTERING$\left(k, \frac{9\pi}{k^2(1+\ell)\log(n)}\right)$ requires $\mathcal{O}\Big(\max\left(k\log n, \log(n)^4\right)\Big)$ rounds. After the execution of the protocol CLUSTERING, w.h.p. any station belongs to a specified cluster and knows its cluster head, which is at a distance of at most $2k$ hops.

Proof: By choosing $p \equiv \frac{9}{k^2(1+\ell)\log(n)}$, we make sure that the disks with radius kr that are centered at the candidate stations achieve a full coverage of the support area X. More precisely, let ξ be the random variable counting the number of candidate stations. The average number of candidate stations is given by $\mathbb{E}(\xi) = np = n\,\frac{9\pi}{k^2(1+\ell)\log(n)}$.

By Chernoff bounds, we know that $\xi = \Theta\left(\frac{n}{k^2\log(n)}\right)$ w.h.p., since $\frac{n}{k^2(1+\ell)\log(n)} \to \infty$. In fact, standard calculus yields

$$\mathbb{P}\left(\frac{1}{3}\mathbb{E}(\xi) \le \xi \le 2\mathbb{E}(\xi)\right) \le 1 - \exp\big(-\mathcal{O}(\mathbb{E}(\xi))\big).$$

Next, by virtue of the result in [25, Thm. 3.2], if $\frac{1}{3}\mathbb{E}(\xi)k^2r^2 > 2.83|X|$ the disks generated by the candidate stations ensure a full coverage of the support area $|X|$ w.h.p.

Then, it is easily seen that the elimination of two "colliding" candidate CHs can be worked out by using the BROADCAST protocol. Similarly, any station which still needs a cluster head is assigned the closest CH in its $2k$-hops neighborhood, by means of the COLLECT protocol.

Finally, CLUSTERING is made of ASSIGNCOLOR and BROADCAST, which require $\mathcal{O}\left(\log n^4\right)$ (cf. [22]) and $\mathcal{O}(k\log n)$ rounds, respectively. Hence, the time complexity of CLUSTERING is clearly $\mathcal{O}\Big(\max\left(k\log n, \log(n)^4\right)\Big)$. ■

C. Learning the neighborhood and local initialization

The aim of the protocol TOTALKNOWLEDGE is to allow each node to "learn" the topology of its i-hops neighborhood, where i is a parameter of the procedure. In order to construct the adjacency matrix of its neighbors, a given node executes the local procedure APPENDTOADJACENCYMATRIX. It works as follows:

- Every node u (with degree d_u) maintains a local list $L(u)$, initialized to

$$L(u) := \text{TMPID}(u) \rightarrow \text{NIL}.$$

- Upon receiving the number of its direct neighbors v_1, v_2,..., v_{d_u}, u updates $L(u)$ to

$$\begin{array}{lccc} L(u) := \text{TMPID}(u) \rightarrow & v_1 & \cdots & v_{d_u} \rightarrow \text{NIL} \\ & \downarrow & \cdots & \downarrow \\ & \text{NIL} & \cdots & \text{NIL}. \end{array}$$

- Next, every neighbor v_1,..., v_{d_u} sends its respective list $L(v_1)$,..., $L(v_{d_u})$ that u appends to its current list and constructs its own neighborhood adjacency matrix.

Clearly, after i steps each participating node can build its own $i \times i$ adjacency matrix, which represents its i-hops neighborhood.
The following procedure TOTALKNOWLEDGE, which runs APPENDTOADJACENCYMATRIX is as follows.

```
1   Procedure TOTALKNOWLEDGE(i :: integer)
2   Begin
3    Each node u, with TMPID(u) and color c(u),
   maintains a list L(u) := TMPID(u);
4    t := i;
5    Repeat (100 × i × log n) times
6      If t > 0 and (TIME mod c(u)) ≡ 0 Then
7        TRANSMIT(L(u), t);
8        t := t − 1;
9      End If
10      Upon receiving a list L do
11    L(u) :=APPENDTOADJACENCYMATRIX(L);
12    End Repeat
13   End.
```

The local initialization protocol LOCALINIT is the combination of the two protocols CLUSTERING and TOTALKNOWLEDGE

```
1   Procedure LOCALINIT(i :: integer)
2   Begin
3     TOTALKNOWLEDGE(i);
4     For each node u belonging to CLUSTER(u)
     LOCALID(u) := rank of u in the sorted
     array of IDs of all nodes in CLUSTER(u);
5   End.
```

D. Paths construction between clusters

Each node u runs TOTALKNOWLEDGE$(4k)$ independently. After what, u owns the adjacency matrix of all its $4k$-hops neighbors. With this information, and the knowledge of all its neighboring clusters, Bellman-Ford algorithm is executed. Every node can thus deterministically build the same routing table between two neighboring clusters; more precisely, between any given pair (s, t) of stations, each within its respective cluster, as described in Fig. 5.

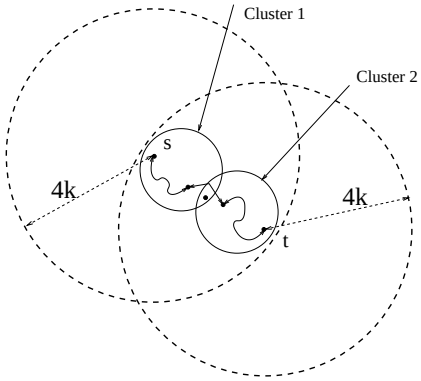

Fig. 5. The choice of $i = 4k$ as parameter of TOTALKNOWLEDGE allows the nodes to construct all paths deterministically, one after the other, between any two neighboring clusters.

Therefore, the fact that all involved nodes do have the same choice of the pair (s, t) is important of course. For example, the first pair of nodes (s, t) between two adjacent clusters may be taken as the two smallest nodes LOCALIDs in both ones. If such an (s, t)-path exists, the Bellman-Ford algorithm executed by the participating stations finds it. Observe that the latter protocol is not runned distributively. Besides, the choice of the parameter $4k$ ensures that all these stations have the right required adjacency submatrix (which size is at most $2k \times 2k$).

E. Gossiping between clusters and main results

Finally, a gossiping algorithm over disjoint paths with length at most $4k$, is performed over the graph of clusters.
As shown in Fig. 6, the communicating process between two neighboring clusters is then worked out along the constructed disjoint paths. In order to synchronize the communication between adjacent clusters, we cut up the time into "phases" that are $4k$ time slots long. Each such phase is actually made of an $\mathcal{O}(k)$ communication delay time: it serves as a kind of frame in the swap-over process from a given path to a next disjoint one. The gossiping algorithm is therefore deterministic, and in each round every node knows exactly whether sleeping or communicating.

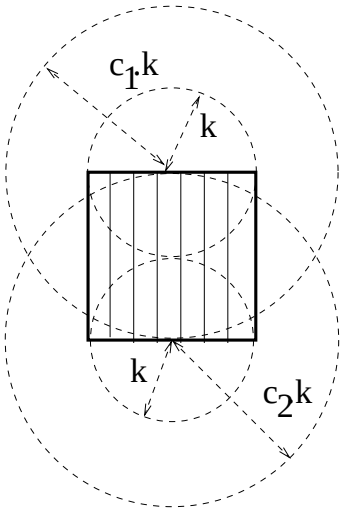

Fig. 6. The square of surface $\mathcal{O}\left(k^2r^2\right)$ and its m regular strips frame, that link two half-covered neighboring clusters for swapping over between the disjoint paths

Lemma 2: Let CLUSTER(u) and CLUSTER(v) be any two adjacent clusters. W.h.p., there exist at least $\mathcal{O}\big(k^2\big)$ disjoint paths between CLUSTER(u) and CLUSTER(v).

Proof: (Sketch)
Let c_1k and c_2k be the hop-radius of two neighboring clusters. Clearly $1 \leq c_1 \leq 2$ and $1 \leq c_2 \leq 2$. As shown in Fig. 6, there exists a square S of surface $|S| = \mathcal{O}\left(k^2r^2\right)$ covering half of each two clusters.
Split S into m regular (rectangular) strips S_i of equal size $|S_i| = \mathcal{O}\big(\frac{k^2r^2}{m}\big)$ $(i \in [1, m])$, and let N_i be the number of stations within each strip S_i.
If $k^2r^2/m \gg 1$, $\mathbb{E}(N_i) = \mathcal{O}\big(\frac{k^2r^2}{m}\big) \gg 1$. Now, by Chernoff bounds, we know that there exist two constants ν_i and μ_i for each $1 \leq i \leq m$, such that

$$\mathbb{P}\left(\nu_i \frac{k^2r^2}{m} \leq N_i \leq \mu_i \frac{k^2r^2}{m}\right) \geq 1-\exp\left(-\mathcal{O}\left(\frac{k^2r^2}{m}\right)\right).$$

Next, fix $i \in [1, m]$ and denote r_{CON}, the transmission range required to have a connected graph inside the strip S_i. Among other results, Penrose proved in [18] that if $N_i/|S_i| = O(1)$,

$$\lim_{N_i \to \infty} \mathbb{P}\left(\pi \frac{N_i}{|S_i|} r_{\text{CON}}^2 - \log\left(N_i\right) \leq \omega\right) = \exp\left(-e^{-\omega}\right).$$

Finally, if we let $m = \mathcal{O}\big(k^2\big)$, then

$$\frac{\log\left(N_i\right) \times |S_i|}{N_i} = \mathcal{O}\left(\log\left(\frac{k^2r^2}{m}\right)\right) = \mathcal{O}(\log\log n).$$

In the present case, the transmission radius satisfies $r^2 = \mathcal{O}(\log n)$, and therefore, any subgraph within S_i is connected with probability greater than $\exp\left(-n^{\Theta(1)}\right)$. Since the number m of strips is at most polynomial in n, it is growing much slower than the above probability of any subgraph in S_i to be connected; and this holds for all $i = 1, 2, \ldots, m$. Hence, w.h.p. the number of disjoint paths between CLUSTER(u) and CLUSTER(v) is at least $\mathcal{O}\big(k^2\big)$, and we are done. ■

As an immediate consequence, we have the following main Theorem 3

Theorem 3: Let n stations be randomly deployed on a support area X with linear size, $|X| = \mathcal{O}(n)$, and assume the radius of transmission of each station to be $r = \sqrt{\frac{(1+\ell)\log(n)|X|}{\pi n}}$.
For any $k \ll \sqrt{\frac{n}{\log n}}$, the initialization of the stations requires $\mathcal{O}(k \times \sqrt{n \log n})$ rounds, with no station being awake for more than $\mathcal{O}\left(\max\left(\frac{\sqrt{n\log n}}{k}, k\log n, \log\left(n\right)^4\right)\right)$ rounds.

Proof: If each cluster is considered as a graph node, the running time of the initialization protocol is $\mathcal{O}\left(k \times D \times \log n\right) = \mathcal{O}\big(k\sqrt{n\log n}\big)$ rounds (where D denotes the hop-diameter of the graph). Swapping over from (disjoint) path to path between adjacent clusters requires that each node is used only every $\mathcal{O}(k^2)$ rounds, and the result follows. ■

Corollary 4: Under the assumptions of Theorem 3, there exists a randomized initialization protocol running in $\mathcal{O}\left(n^{3/4}\log\left(n\right)^{1/4}\right)$ rounds, with no station being awake for more than $\mathcal{O}\left(n^{1/4}\log\left(n\right)^{3/4}\right)$ rounds.

IV. CONCLUSION

In the present paper, a performing and energy-efficient algorithm for the initialization problem is designed. Its running time, as well as the awake time per station are both broadly sublinear. More precisely, the time complexity of our protocol achieves $\mathcal{O}\left(n^{3/4}\log\left(n\right)^{1/4}\right)$ rounds, with no station being awake for more than $\mathcal{O}\left(n^{1/4}\log\left(n\right)^{3/4}\right)$ rounds.
It is also worth to emphasize the fact that choosing $k = \mathcal{O}(1)$ yields an almost time optimal protocol. In such a case indeed, the running time shrinks to $\mathcal{O}\big(\sqrt{n\log n}\big)$, whereas the easier broadcast problem requires at most $\Omega\left(\sqrt{\frac{n}{\log n}}\log\log n\right)$ rounds [6], [11]. Hence, our result is at most $O\big(\log n/\log\log n\big)$ far from optimality.

Finding the lower-bound on the awake time per station for the initializing stations in a random radio network is an open challenging problem. Furthermore, an even more challenging open problem remains of course the design and analysis of an initialization algorithm which could reach the latter lower-bound while keeping a nearly optimal time complexity.

REFERENCES

[1] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. Wireless sensor networks: a survey. *Computer Networks* 38: 393-422, 2002.

[2] Alon, N., Bar-Noy, A., Linial, N. and Peleg, D. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43: 290-298, 1991.

[3] Bar-Yehuda, R., Goldreich, O. and Itai, A. Efficient Emulation of Single-Hop Radio Network with Collision Detection on Multi-Hop Radio Network with no Collision Detection. *Distributed Computing*, 5: 67-71, 1991.

[4] Bar-Yehuda, R., Goldreich, O. and Itai, A. On the Time-Complexity of Broadcast in Multi-Hop Radio Networks: An Exponential Gap between Determinism and Randomization. *Journal of Comp. and Sys. Sciences*, 45: 104-126, 1992.

[5] Chlebus, B. Randomized Communication in Radio Networks Chapter in *"Handbook of Randomized Computing*," Panos M. Pardalos, Sanguthevar Rajasekaran, John H. Reif, and Jose D.P. Rolim (Eds.), Kluwer Academic, New York, 2001, vol. I, pp. 401-456.

[6] Chrobak, M., Gasieniec ,L. and Rytter, W. Fast Broadcasting and Gossiping in Radio Networks. *Proc. IEEE F. of Comp. Sci. (FOCS)*, 2000.

[7] Gupta, P. and Kumar P. R. Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications: a volume in honor of W. H. Fleming, W. M. McEneaney, G. Yin and Q. Zhang*, Birkhauser, Boston, 1998.

[8] Hall, P. *Introduction to the Theory of Coverage Processes*. Birkhäuser, Boston, 1988.

[9] Hayashi, T., Nakano, K. and Olariu, S. *Randomized Initialization Protocols for Packet Radio Networks*, in S. Rajasekaran, P. Pardalos, B. Badrinath, and F. Hsu, Eds., Discrete Mathematics and Theoretical Computer Science, SIAM Press 2000, 221–235.

[10] Kushilevitz, E. and Mansour, Y. An $\Omega(D \log(N/D))$ Lower Bound for Broadcast in Radio Networks SIAM Journal on Computing, Vol. 27, 702-712, 1998.

[11] Liu, D. and Prabhakaran, M. *On Randomized Broadcasting and Gossiping in Radio Networks* in Proceedings of COCOON'02, Lecture Notes in Computer Sciences, Vol. 2387, 340-349, 2002.

[12] Meester, R. and Roy, R. *Continuum Percolation*. Cambridge University Press, Cambridge, 1996.

[13] Nakano, K., Olariu, S. *Leader election protocols for radio networks.* In Handbook of wireless networks and mobile computing. pages 219-242. John Wiley & Sons, Inc. (2002).

[14] Nakano, K. and Olariu, S. Randomized Initialization Protocols for Ad-hoc Networks. *IEEE Transactions on Parallel and Distributed Systems* 11: 749-759, 2000.

[15] Nakano, K. and Olariu, S. Energy-Efficient Initialization Protocols for Radio Networks with no Collision Detection. *IEEE Transactions on Parallel and Distributed Systems* 11: 851-863, 2000.

[16] Penrose, M. D. The longest edge of the random minimal spanning tree. *Annals of Applied Probability*, 7: 340-361, 1997.

[17] Penrose, M. D. A strong law for the largest nearest-neighbour link between random points. *Journal of the London Mathematical Society*, 60: 951–960, 1999.

[18] Penrose, M. D. On k-connectivity for a geometric random graph. *Random Structures & Algorithms*, 15: 145-164, 1999.

[19] Penrose, M. D. *Random Geometric Graphs*. Oxford Studies in Probability, 2003.

[20] Perkins, C. E. *Ad Hoc Networking*. Addison-Wesley, 2001.

[21] Ravelomanana, V. Extremal Properties of Three Dimensional Sensor Networks with Applications. *IEEE Trans. on Mobile Computing*, 3: 246-257, 2004.

[22] Ravelomanana, V. Optimal Initialization and Gossiping Algorithms for Random Radio Networks. *to appear in IEEE Trans. Parallel and Distributed Systems*.

[23] Santi, P. and Blough D. M. The Critical Transmitting Range for Connectivity in Sparse Wireless Ad Hoc Networks. *IEEE Trans. Mob. Comp.*, 2: 1-15, 2003.

[24] Xu, Y. An $\mathcal{O}(n^{1.5})$ Deterministic Gossiping Algorithm For Radio Networks. *Algorithmica*, 36: 93-96, 2003.

[25] Muthukrishman, S.and Pandurangan, Gopal. The Bin-Covering Technique for Thresholding Random Geometric. *Proc. of ACM-SIAM'06 symposium on Disccrete algorithms*, p. 989-998, 2005.