HAL
open science

An energy-efficient initialization algorithm for random radio networks

Binh Thanh Doan, Christian Lavault, Stephan Olariu, Vlady Ravelomanana

To cite this version:

Binh Thanh Doan, Christian Lavault, Stephan Olariu, Vlady Ravelomanana. An energy-efficient initialization algorithm for random radio networks. 2007. hal-00153106

HAL Id: hal-00153106

https://hal.science/hal-00153106v1

Preprint submitted on 8 Jun 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An Energy-Efficient Initialization Algorithm for Random Radio Networks

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

***Abstract*— A radio networks 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. 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 that the nodes have no structural knowledge. To solve the initialization problem, we propose an energy-efficient randomized algorithm running in at most $\mathcal{O}(MMMM)$ time slots, with no station being awake for more than $\mathcal{O}(NNNN)$ 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 in the network can move; so the topology is unstable. For this reason, we must refrain from assuming too much about the knowledge of the network topology in the design of protocols. In this

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 – Nord, France. E-mail : {vlad,lavault}@lipn.univ-paris13.fr.

work, the nodes are assumed not to have any initial structural information of the network (no topological knowledge, s.a. the size of the network for example). On the other hand, every node knows the number n of participating stations and the measure $|X|$ of the surface X where they are randomly dropped.

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 $\mathcal{O}(n^{1/2}\log n^{1/2})$ and $\mathcal{O}(n^{1/3}\log n^{2/3})$ time slots (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 $\mathcal{O}(NNNN)$ rounds with no station being awake for more than $\mathcal{O}(MMMM)$ 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 of order $\mathcal{O}(D\log n) = \mathcal{O}(D\Delta)$, where Δ is the maximum degree of the underlying network and D denotes its eccentricity (hop-diameter). Indeed it was shown in [11] that, under the same settings, the easiest broadcasting problem requires $\Omega(D\log\log n)$ rounds.

Outline of the paper. The remainder of this extended abstract is organized as follows. Section 2 discusses the principal steps, directions to solve our problem. In Section 2, we present the main steps PREPARATION, CLUSTERING, LOCAL INITIALIZATION, GLOBAL INITIALIZATION of our result. The ideas behind these steps can be briefly described 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. Then, we apply a divide-and-conquer principle: we cluster the graph (CLUSTERING) and each cluster is locally initialized (LOCAL INITIALIZATION). Next, a global initialization is performed by means of a random walk between clusters. Such a walk induces a random ordering between clusters, which makes it possible to initialize the whole network as soon as all the clusters are visited. All details, viz. the designs and analysis of our algorithms are then given in Section 3. Finally, Section 4 provides some concluding remarks as well as open problems.

II. MAIN STEPS

A. Fundamental characteristics of the network

In our settings, the n nodes are deployed randomly in a given a square X of size $|X|$ with $n = \mathcal{O}(|X|)$. Each node has a transmission radius of $r = \sqrt{\frac{(1+\ell)\log(n)|X|}{\pi\, n}}$. Under this scenario, we know that with high probability[2], the underlying reachability graph satisfies the following main characteristics.

- 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.

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

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

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 by assumption 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 draw the same TMPID (see also [22]).

Moreover, collisions can also 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 invocation, any two nodes at a distance of at most 2-hops from each other are assigned two distinct colors (or codes). Thus, 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 objective of this step is to design a randomized algorithm that partition 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. 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 invocation of the clustering protocol, each cluster can be initialized locally.

D. Local initialization

To initialize each cluster locally, a protocol called GOSSIP is used. The idea is very similar to those in [22]. Our protocol is executed distributively by all the stations in all the clusters. Every node in the network transmits its TMPID to all other stations. Our 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 neighbours. Since the coloration algorithm uses $\mathcal{O}(\log n)$ colors, after $\mathcal{O}(k \log n)$ time slots, all stations know the TMPIDs of all the others in their cluster. At the end, 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 communications 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 most crowded paths), we have to find out as many paths as possible and swap 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 realized via the paths between clusters (by swaps from one to one).

The figures 1 and 2 below depict and summarize briefly the main steps of the initialization protocol.

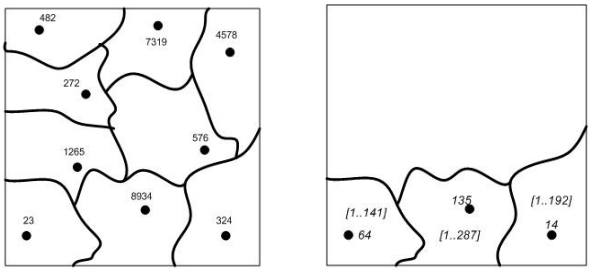

Fig. 1. Division of the graph into disjoint clusters and local initialization of each cluster. In the figure on the right above, 3 clusters are initialized with integers ranging from 1 to 141, from 1 to 287, and from 1 to 192, respectively.

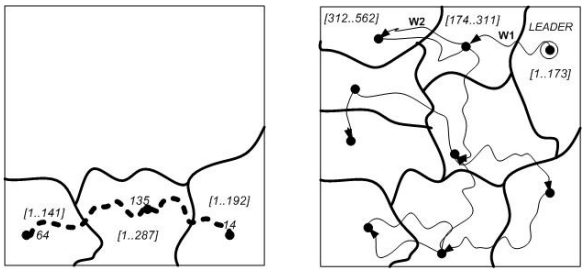

Fig. 2. After the paths construction (dashed lines) between clusters, the global initialization is just executed by means of the gossiping algorithm performed via these paths.

III. Detailed Algorithms and Analysis

A. Coloring, broadcasting and gossiping

In this paragraph, we are concerned with 3 basic protocols which will be frequently used and discussed throughout the remainder of the paper. First, the protocol AssignColor is executed (see the design in [22, Section VI]). The running time of AssignColor is $\mathcal{O}\left(\log n^4\right)$ 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 probability tending to 1 as $n \to \infty$.

Once it is well-colored, the network is collision-free and we are now ready to design the broadcasting protocol, whose pseudo-code is given in Fig. 3.

It is easily seen that such protocol works in $\mathcal{O}(k \log n)$ time-slots (under the condition that the randomized coloring algorithm succeeds with probability 1, i.e, if it does not err).

Based upon Broadcast, one can design a gossiping algorithm. In the gossiping problem, the task is to disseminate 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.

Procedure BROADCAST(msg :: message, k :: integer)
Begin
 Repeat $(100 \times k \times \log n)$ **times**
  For a node u of color **c(u)**, upon receiving a message of the form $\langle msg, k\rangle$ **Do**
   If (TIME mod c(u)) $\equiv 0$ and $k > 0$ **Then**
    BROADCAST($msg,\ k-1$)
 End Repeat
End.

Fig. 3. The BROADCASTING algorithm.

Procedure Gossip(k :: integer)
Begin
 Repeat $(100 \times k \times \log n)$ **times**
  For each node u with initial message $msg(u)$ and color **c(u)** upon receiving any message of the form $\langle msg,\ k\rangle$ **Do**
   If (TIME mod c(u)) $\equiv 0$ and $k > 0$ **Then**
    $msg := append(msg, msg(u))$;
    Transmit(msg) ;
    Gossip($k-1$) ;
 End Repeat
End.

Fig. 4. The GOSSIPING algorithm.

Since there are $\mathcal{O}(\log n)$ colors and k steps, Gossip(k) runs in the same execution time as Broadcast(k): $\mathcal{O}(k \log n)$.

B. Random clustering

In order to apply a divide-and-conquer algorithm, a CLUSTERING protocol is used, which works as follows. First, each station chooses to be a candidate cluster head (CH) with a certain probability p (to be 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, some 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 by each CH, which executes a protocol with a specific message: each orphan can actually choose the nearest CH in its $2k$-neighborhood :

Procedure Collect(j :: integer)
Begin
 For each node u **Do**
 If u is a cluster head **Then**
  Repeat $(100 \times j \times \log n)$ **times**
   If (TIME mod c(u)) $\equiv 0$ **Then**
    Transmit(TmpID,1)

End Repeat
Else
CLUSTER(u):=NIL, distance := ∞;
Upon receiving a message of the form $\langle$TMPID, radius$\rangle$
If distance $>$ radius **Then**
CLUSTER(u):=TMPID, distance := radius;
Repeat $(100 \times j \times \log n)$ **times**
If (TIME mod c(u)) $\equiv 0$ **Then**
TRANSMIT(CLUSTER(u),distance+1);
End Repeat
End If
End Else
End.

The above protocol is designed to collect the orphan nodes, i.e., the nodes that need a cluster head.

The pseudo-code of the CLUSTERING protocol is therefore given below.

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

We have the following result.

Theorem 1: The complexity of CLUSTERING$\left(k, \frac{9\pi}{k^2(1+\ell)\log(n)}\right)$ is $\mathcal{O}\left(\max(k\log n,\ \log(n)^4)\right)$ time slots. After CLUSTERING, with high probability any station belongs to a specified cluster and knows its cluster head, which is within 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 $k \times r$ 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 using 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 just yields

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

Now, we use the result of Muthukrishman and Pandurangan [25, Theorem 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. Next, 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 consists of ASSIGNCOLOR and BROADCAST, whose running times are $\mathcal{O}(\log n^4)$ (cf. [22]) and $\mathcal{O}(k\log n)$, respectively. Hence, the complexity of CLUSTERING is clearly $\mathcal{O}\left(\max\left(k\ \log n, \log n^4\right)\right)$ time slots. ■

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 node uses a local procedure called APPENDTOADJACENCYMATRIX. It works as follows:

- Every node u maintains a local list $L(u)$, initialized to
$$L(u) := \text{TMPID}(u) \to \text{NIL}.$$
- Upon receiving the number of its direct neighbors $v_1, v_2, \ldots, v_{d_u}$, u updates $L(u)$ which becomes
$$L(u) := \text{TMPID}(u) \to \begin{array}{ccc} v_1 & \cdots & v_{d_u} \to \text{NIL} \\ \downarrow & \cdots & \downarrow \\ \text{NIL} & \cdots & \text{NIL}. \end{array}$$
- Next, its neighbors $v_1, \ldots v_{d_u}$ send their respective lists, namely $L(v_1), \ldots L(v_{d_u})$, that u appends to its current list and constructs its 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 procedure TOTALKNOWLEDGE running APPENDTOADJACENCYMATRIX is as follows.

Procedure TOTALKNOWLEDGE(i :: integer)
Begin
Each node u with TMPID(u) and color $c(u)$ maintains a list $L(u) :=$ TMPID(u);

$t := i$;
Repeat $(100 \times i \times \log n)$ **times**
  If $t > 0$ and **(TIME mod c(u))** $\equiv 0$ **Then**
    $\textsc{Transmit}(L(u), t)$;
    $t := t - 1$;
  End If
  Upon receiving a list L do
  $L(u) := \textsc{AppendToAdjacencyMatrix}(L)$;
End Repeat
End.

LocalInit, the local initialization protocol, is the combination of the previous algorithm and the protocol Clustering.

Procedure $\textsc{LocalInit}(i$ :: integer)
Begin
  $\textsc{TotalKnowledge}(i)$;
  For each node u belonging to $\textsc{Cluster}(u)$ $\textsc{LocalID}(u) :=$ rank of u in the sorted array of IDs of all nodes in $\textsc{Cluster}(u)$;
End.

D. Paths construction between clusters

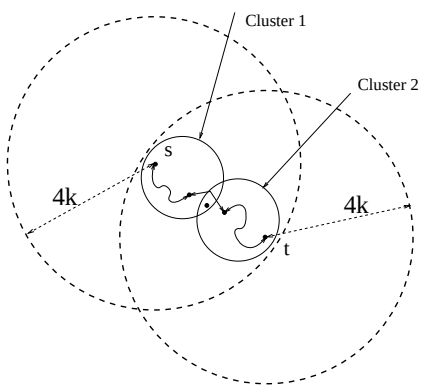

Fig. 5. The choice of $i = 4k$ as parameter of TotalKnowledge allows the nodes to construct all paths deterministically, one after the other, between any two neighboring clusters.

Each node u runs $\textsc{TotalKnowledge}(4k)$ independently. After what, u owns the adjacency matrix of all its $4k$-hops neighbors. With this information, and knowing the topology of all its neighboring clusters, Bellman-Ford algorithm is runned. Every node can thus deterministically build the same routing table between any given pair (s, t) of stations within two neighboring clusters, as shown in Fig. 5. So, having the same choice of the pair (s, t) is important for all involved nodes; e.g., the first pair (s, t) between two given clusters may be taken as the smallest two LocalIDs in both ones. If such a (s, t)-path exists, the Bellman-Ford algorithm executed by the participating stations will find it. Observe that the latter algorithm is not runned distributively. The choice of the parameter $4k$ makes it sure that these stations have the right required adjacency submatrix (which size is at most $2k \times 2k$).

E. Gossiping between clusters and main results

Using the paths of length at most $4k$, a gossiping algorithm is now runned on the graph of clusters. To synchronize, each TIME-SLOT is divided into $4k$ time-slots. And in each BIGTIME-SLOT, (en gros les frames , i.e. les O(k) time-slots qui permuttent) TITAN path between 2 adjacent clusters are used to communicate between clusters. The gossiping algorithm is therefore deterministic and in each time slot, every node knows exactly what to do either sleeping or communicating.

Lemma 2: Let $\textsc{Cluster}(i)$ and $\textsc{Cluster}(j)$ be two adjacent clusters. W.h.p., there exists at least $\mathcal{O}(k^2)$ disjoint paths between $\textsc{Cluster}(i)$ and $\textsc{Cluster}(j)$.

Proof: Let $c_1 k$ and $c_2 k$ be the hop-radius of the 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 $|S| = \mathcal{O}(k^2 r^2)$ between the two clusters.

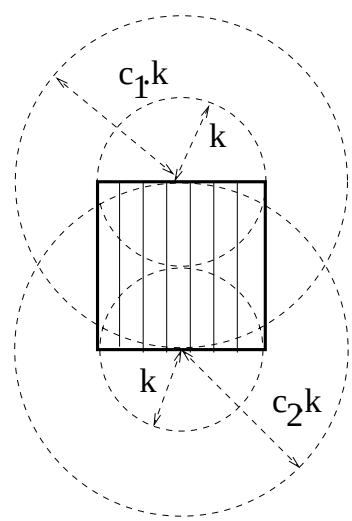

Fig. 6.

Split S into m REGULAR LAMELLES of equal size $|S_i| = \mathcal{O}\left(\frac{k^2 r^2}{m}\right)$, $i \in [1, m]$. Let N_i be the number of stations in each LAMELLE S_i. If $k^2 r^2 / m \gg 1$ so that $\mathbb{E}[N_i] = \mathcal{O}\left(\frac{k^2 r^2}{m}\right) \gg 1$, using Chernoff bounds, we know that there exists constants ν_i and μ_i such that

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

Fix $i \in [1, m]$. Denote by r_{CON} the transmission range required to have a connected graph inside S_i. Among other results, Penrose [18] proved (with our notations) that if $N_i/S_i = O(1)$ then if

$$\lim_{N_i \to \infty} Pr\left[\pi \frac{N_i}{|S_i|} r_{\text{CON}}^2 - \log(N_i) \leq \omega\right] = \exp\left(-e^{-\omega}\right). \tag{1}$$

Now, suppose that $m = \mathcal{O}(k^2)$. Thus,

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

In our case, since the transmission range satisfies $r^2 = \mathcal{O}(\log n)$, the subgraph inside S_i is connected with probability greater than $\exp\left(-n^{\Theta(1)}\right)$. Since the number of LAMMELES is at most polynomial in n (THIS GROWTH is much less than so the proba is OK!).... TITAN HELP, please. ■

As an immediate consequence, we have:

Theorem 3: Given n randomly deployed stations on a support area X of size $|X| = \mathcal{O}(n)$. Suppose that their radius of transmission is set to $r = \sqrt{\frac{(1+\ell)\log(n)|X|}{\pi n}}$. For any $k \ll \sqrt{\frac{n}{\log n}}$, the initialization runs in $\mathcal{O}(k \times \sqrt{n \log n})$ rounds with no station being awake for more than $\mathcal{O}\Big(\max\left(\frac{\sqrt{n\log n}}{k}, k\log n, \log(n)^4\right)\Big)$ time-slots.

Proof: The running time is given by SI chaque cluster est pris comme un sommet ... $\mathcal{O}(k \times D \times \log n) = \mathcal{O}(k\sqrt{n \log n})$. By swapping between the paths between adjacent clusters, each node is used every $\mathcal{O}(\log n)$ time-slots. bla bla bla ■

Corollary 4: Under the same settings as in Theorem 3, there exists a randomized initialization algorithm running in $\mathcal{O}(n^{3/4} \log(n)^{5/4})$ time-slots with no station being awake for more than $\mathcal{O}(n^{1/4} \log(n)^{3/4})$ rounds.

IV. Conclusion

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*, pp. 989–998, 2005.