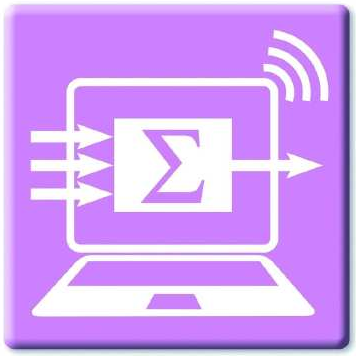


Connectivity Awareness in Networked Robotics System

Van Tuan Le[1,2]
le@ensm-douai.fr

Noury Bouraqadi[1]
bouraqadi@ensm-douai.fr

Serge Stinckwich[2]
serge.stinckwich@info.unicaen.fr

Victor Moraru[2]
moraru@auf.org

1: Dept. I.A, Ecole des Mines de Douai.
59500 Douai, France

2: MSI/IFI, Hanoi.
Vietnam

Technical Report 20081001
1er octobre 2008

Dpt. Informatique et Automatique (DIA)
Ecole des Mines de Douai
941 rue Charles Bourseul - B.P. 838
59508 Douai Cedex
France

Contents

Abstract

Maintaining the network connectivity during the mission of robots in Mobile Multi-Robot Systems (MRSs) is a key issue in many robotics applications. In our view, the solution to this problem consists of two main steps: *(i) making robots aware of the network connectivity; and (ii), making use of this knowledge in order to plan robots tasks without compromising the connectivity*. In this paper, we view the network connectivity as an abstraction that is independent from application issues. With respect to (i), we propose a new distributed algorithm that will be executed on individual robots to build and maintain the connectivity-awareness. The correctness and theoretical analysis, as well as the simulation results of the proposed algorithm are given. For illustrating (ii), first we show how our solution allows checking the robustness of network connectivity more efficiently than existing works; and second, we present an application of using this awareness in distributed robot motion control to preserve the robot network connectivity.

keywords: Mobile Ad Hoc Network, Multi-Robot Systems, Distributed Robot Motion Control, Dynamic Graph, Connectivity Maintenance.

1 Introduction

The use of MRSs is promising in applications such as rescue operations after natural disasters like earthquakes. In such situations, a group of autonomous robots has to collaboratively perform a mission whose success depends on communication between individuals. Many studies have led to the conclusion that even the exchange of a small amount of information improves the performance of multi-robot systems in certain kinds of tasks [Mac91, BA94].

In this paper, we are interested in communication between robots in a team relying on Mobile Ad Hoc Network (MANET) technologies [Per01]. In such networks, a robot is not only an "ordinary" networked node, but also a router that relays messages for their neighbors. The communication between robots which are not neighbors can thus take place through consecutive intermediate relaying nodes. Furthermore, robots are able to detect some neighborhood-related events (e.g. the appearance or disappearance of neighbors) thanks to the underlaying networking services. Though useful, these information are not enough to enable robots to plan their motion while preserving their *network connectivity* during the mission. In our vision, solutions for this problem should consist of two main steps: *(i) making robots acquire a sufficient knowledge on the network connectivity; and (ii), exploiting this knowledge in order to maintain the best the network connectivity while performing other tasks.* We further argue that the first step can be viewed as an application-independent abstraction. That is, the awareness of the network connectivity should be provided to robots as a basic networking service like routing in MANETs for example.

Thereafter, we present our solution for the step (i); in which, we attempt to make autonomous robots in a MRS individually aware of the network connectivity. The work presented here can be considered complementary to the mobile networking research field. And then for illustrating the step (ii), we present the use of the connectivity awareness in two applications: verifying the robustness of a wireless network, and distributed robot motion control. The rest of the paper is

organized as follow. In section 2, we present some needed definitions about graph theory in the context of MRS and set up the background for upcoming discussions. Section 3 introduces a new distributed algorithm to make robots in MRS connectivity-aware. The maintenance of this robots' awareness in the presence of the mobility is then described in section 4. The concept of connectivity is further extended to robust connectivity in section 5. This extension includes also an efficient distributed algorithm for critical nodes detection. In section 6, a use of connectivity awareness for the maintenance of network connectivity in multi-robot exploration is presented. Section 7 presents related works; and finally, section 8 concludes with a summary of the benefits of our proposal and future work.

2 Awareness of Network Connectivity

As an example for illustrating our approach to maintaining the connectivity, first consider a leader-follower MRS like the one in the figure 1. The team is sent out to explore and build the map of some unknown area. To speed up the exploration, robots in team need to spread out as large as possible in the ground. On the other hand, they must keep in touch with each other, and particularly with the leader (robot 1), so that they can share the map of the explored area, and thus avoid overlaps between them. An efficient exploration algorithm has to find out a good compromise between these two conflicting constraints. For that, one has to deal with the question: given the limited communication range of robots, how can robots individually choose a move toward a target while keeping in touch with the other teammates.

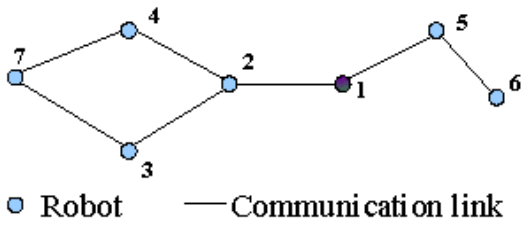


Figure 1: A Networked Robotics System.

The basic idea in maintaining the network connectivity in such robotic application is that: each robot i in the team, while performing its task, has to keep in touch with *at least* one neighboring robot j from which we can find a set of consecutive relaying robots toward the leader (robot 1). Concretely, in the system shown in figure 1, robot 5 and robot 2 have to maintain their connections with robot 1, and robot 6 has to stay in touch with robot 5. Robot 3 and robot 4 should move around in such a way that the links between them and robot 2 will not be broken. For robot 7, there are two different paths to robot 1, it needs to maintain *at least* one link with either robot 3 or robot 4. So robot 7 has more choice to move while taking the connectivity into account. If robots are all successful as such, then the connectivity of the whole system will be ensured.

We model a networked robots system by an *undirected* graph $G = (V, E)$, where V is the set of robots in the network, $E = V \times V$. There is an edge $e = \{u, v\} \in E$ if and only if u

and v can *mutually* receive each other's transmission, i.e. the link between them is bidirectional. In this case, we say u and v are neighbors, and the edge is also referred as a *communication link* between the two robots. Note that the graph is dynamic i.e can change in time as nodes are moving. The term *node* is used to indicate a robot, or vice versa interchangeably.

Definition 1 (Communication path)**.** In the graph G, a *loop-free* sequence of nodes taking part in the process of relaying data from u to v or symmetrically from v to u is called a *communication path* $p(u, v)$ (or simply a path hereafter).

We use a vector-like representation to denote a path $p(u_1, u_n) = (u_1, u_2, ..., u_n)$. For some node k, $k \in p$ or $k \notin p$ specifies that the path p includes k or not, respectively. By definition, any edge $e \in E$ is also a path.

Definition 2 (Connected graph)**.** A graph G is said to be connected if and only if for any $u, v \in V$, there exists a path $p(u, v)$.

In our work, instead of the term "any" in the above definition, we take a "fixed" node and term it the *reference node*. The definition 2 is equivalently restated as follow.

Definition 3 (Referentially Connected Graph)**.** Given a reference node $r \in V$, the graph G is called referentially connected if and only if for any other node $u \in V$, there exists a path $p(u, r)$.

Definition 4 (Access Robot and Access Path)**.** Given the reference robot r, and two different robots $u, v \in V$, v is called an access robot for u if and only if there exist an edge $e\{u, v\} \in E$[1] and there exist a path $p(v, r)$ such that $u \notin p$. We call p an *access path*.

We coin the knowledge (stated by the definitions 3 and 4) by the term *awareness of the network connectivity* or simply *connectivity awareness*.

3 Making Networked Robots Connectivity Aware

A connectivity-awareness for a given robot is materialized as a *connectivity table* (or table for short if there is no confusion) containing a set of access paths. These paths represent a partial view of the network connectivity. For example, the table of robot 7 in the network in figure 1 should have two access paths $(1, 2, 4)$ and $(1, 2, 3)$ corresponding to two access robots 3 and 4. Based on this knowledge, robots know which neighbors they should depend on for maintaining the connectivity with the whole network. For instance robot 5 knows that it needs to maintain the connection with robot 1, but not with robot 6 because there is one path refering to robot 1 as the sole access robot. On the contrary, robot 6 must keep in touch with robot 5. This section presents our algorithm to build the connectivity table. For the sake of simplicity and in order to be comprehensive, we make assumption that the connectivity should not change during the execution of the algorithm. In fact, all the presented results can be covered without this assumption as in section 4, the maintenance of the connectivity table in the presence of the

[1] u and v are neighbors

mobility is introduced; and in section 6, the simulations with moving robots also validate our approach.

Assume that at the beginning of a mission, robots are close to each other and form a network; therefore, they can communicate and rely on MANET routing protocol for message transmission. We also assume that a message sent by a node is received correctly within a finite period of time (a *step*) by *all* its neighbors, and that every node knows its ID, and IDs of all its neighbors. The main concerns in making robots network connectivity-awareness now turn out to be the problem of selecting the reference robot, and building the tables.

Choosing a reference node can be application-dependent and might involve multiple criteria such as the energy level, the number of neighbors, hardware requirements, etc. In some situations the choice might be easier than others, such as for leader-follower systems, the leader is a good candidate for becoming the reference. Or when the robot team need to maintain the connectivity with a base station, then the latter will be chosen to be the reference node naturally. Another option to generalize the approach is to employ a market-like bidding mechanism to select the node that will be the reference.

Once chosen, the reference robot will broadcast to all its one-hop neighbors a `New-Access-Path` message, which encodes the new access path. In the case of the reference node, the access path is composed of its own ID. The access path also refers to the message's sender as the access robot.

3.1 Basic Messages Forwarding

Algorithm 1: Algorithm to be executed upon the reception of a `New-Access-Path` message – a straightforward version.

Input: The `New-Access-Path` Message M
Output: The connectivity table T of robot is updated

```
1 begin
2     p ← the access path in M;
3     if this.id() ∉ p then
4         add the path p to the connectivity table T;
5         update and forward the message to all neighbors;
6     end
7 end
```

On receiving a `New-Access-Path` message, a robot extracts the access path encoded in the message's content (see algorithm 1). By mean of this information, robot is able to check whether this message has been already received by itself or not. This check step ensures that the path is loop-free. If the message has been already processed, it is simply ignored. Otherwise, robot will add the path to its table and re-broadcasts a `New-Access-Path` message (with its own ID added to the access path sent along with the message), to its neighbors. Hence the

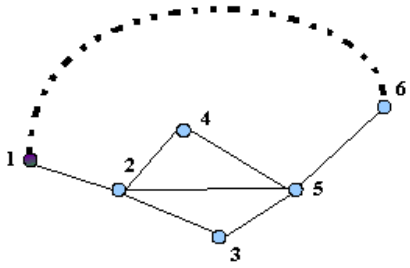


Figure 2: A network configuration that results in an incomplete access robots list for robots 2. Robot 1 is the reference node.

message incrementally spreads out in the network. As long as nodes are reachable from the reference node, they will receive the message and build up their own access table.

3.2 Optimized Messages Forwarding

In the basic version of the algorithm, robots will retransmit all the non-acyclic paths they receive. However, the main objective is to build as complete as possible the list of all access robots among the neighboring ones for individual robots, and not to find and maintain every path toward the reference node. The paths are stored to deal with the mobility in MRSs (to be presented in section 4). This remark leads to a significant reduction in number of forwarded message. First, we notice that after forwarding the first path a robot receives, then all its neighboring robots except the one who has just sent the message, will proceed and consider this robot as one of their access robot.

Because robot know that after the first time it forwards a message, there is only one robot – the sender s of the first message that robot has just received, has not yet registered it as an access robot. The selection of path to forward since then is based on the criteria that a path does not go through s. Therefore, if a robot can become an access node for all its neighbors, it needs to forward only two messages. In this new version (algorithm 2), robots still store all the paths they receive, but forward at most two paths to help their neighbors to build the table as complete as possible.

With a network configuration depicted in figure 2, the algorithm 2 will build a connectivity table stored on each robot that looks like in the table 1.

As seen from the table of robot 3 (table 1(b)), the first time when robot 3 receives the path $(1, 2)$ from robot 2, it forwards this path again. Since then, when it receives two paths $(1, 2, 5)$, and $(1, 2, 4, 5)$ retransmitted by robot 5, because these two paths both go through robot 2, which cannot consider robot 3 as access robot; thus, robot 3 just stores this path in its table without continuing the retransmission.

3.3 Algorithms Analysis

We now prove the correctness of the network initialization procedure depicted in algorithm 1 and 2. We have the following propositions.

Algorithm 2: Algorithm to be executed upon the reception of a `New-Access-Path` message – the optimized version.

Input: The `New-Access-Path` Message M
Output: The connectivity table T of robot is updated

```
1  begin
2    p ← the access path in M;
3    if this.id() ∉ p then
4      add p into the connectivity table T;
5      if already forwarded two messages then
6        return;
7      end
8      if this is the first time robot forwards a message
       OR(firstAccessRobot.id() ∉ p) then
9        firstAccessRobot ← M's sender ;
10       mark the path in the connectivity table as forwarded;
11       update and forward the message to all neighbors;
12     end
13   end
14 end
```

Proposition 1. The process of initializing the network is loop-free, and will terminate within l steps, where l is the length in term of hop-count of the longest access path in the network.

Proof. Loops occur when a message that has been treated by a robot i gets back to this one, and it still proceeds with it. However, line 3 of algorithm 1 (and also in algorithm 2) ensures that loops are filtered out.

The process of network initialization started by the reference robot and spreads out in the network step by step. After k steps, nodes that are *k-hop* away from the reference node are able to build up paths composed of $k-1$ intermediate nodes. As there is no loop in its execution, the procedure will terminate within l step, where l is the length in term of hop-count of the longest access path. □

Proposition 2. The message complexity of the algorithm 2 is $2(n-1)+1$, and there are $2d$ paths in the connectivity table, where d is the number of robot's neighbors.

Proof. The proof is trivial and can be drawn directly from the pseudo-code of the algorithm 2. Because each of $n-1$ robots sends out at most two messages, and the reference robot sends one message. There are at most totally $2(n-1)+1$ messages – and this is the upper bound for the number of messages. In return, a robot can receive at most 2 messages from one neighbor with one path to store, thus the size of the connectivity table is at most $2d$. □

Table 1: Connectivity tables of some robots in figure 2.

(a) robot 2's

Access Robots	Access Paths	Retransmitted
1	(1)	yes

(b) robot 3's

Access Robots	Access Paths	Retransmitted
2	(1, 2)	yes
5	(1, 2, 5)	no
5	(1, 2, 4, 5)	no

(c) robot 4's

Access Robots	Access Paths	Retransmitted
2	(1, 2)	yes
5	(1, 2, 5)	no
5	(1, 2, 3, 5)	no

(d) robot 5's

Access Robots	Access Paths	Retransmitted
2	(1, 2)	yes
3	(1, 2, 3)	yes
4	(1, 2, 4)	no
6	(1, ..., 6)	no

3.4 Simulation and Discussion

The straightforward algorithm (algorithm 1) builds a complete list of the access robots for every non-reference robots in the network. However, it issues a huge amount of messages: for a network of n robots, the message complexity is of $O(n!)$ in the worst case (c.f. see the proposition 7 in appendixe 8) where the network graph is complete. Although the worst case almost never happens in practice, this naive approach is likely practically infeasible for highly densed network. In a simulation with a network composed of 20 robots, the average number of robot neighbors is 4, this algorithm had issued 1,072,254 messages; that mean each robot had to deal with about 53,000 messages on average. Therefore, this version is applicable only for very small network.

In the optimized version, the number of messages travelling in the network is significantly reduced, and has message complexity of $O(2n)$. Nevertheless, this precious reduction comes at the price: the access list of robots is not ensured to be complete for some cases. For example in the figure 2, robot 5 might receive and forward two paths $(1, 2)$ and $(1, 2, 3)$ before the reception of any path through robot 6 from the other direction. This results in no any path following the long path (represented by the curved-dot line) reaches robots 3 or 4, and these two robots forward only one path, making the access robot list of robot 2 incomplete.

In order to find out the number of robots in network whose the access list is not completed

Table 2: Statistics of number of robots whose the access list is not complete.

Statistics Value	Network Size					
	11	15	20	50	100	500
max	0	5	6	11	17	40
min	0	0	0	0	0	0
median	0	0	0	0	2	10

(referred hereafter incomplete nodes for short), we realized series of simulations for random networks with various size (c.f. table 2), and different network density (ranging from 3 to 18).

As can be seen from the table 2, even for large network (with hundreds nodes) the ratio of the incomplete nodes on average (calculated by the median statistics function) is much lower than 10%. Furthermore, most of them lack only one robot in the access list. Therefore, for many purposes (like the application in section 6), the list built by the algorithm 2 is sufficient. Although, a remedy for this shortcoming is also proposed in section 5.

4 Dealing with the Dynamic in the Network Topology

Since the environment is subject to change, and that the robots move during their mission, the network topology can change over time. Besides, the reference robot or any other robot fails to work due to various reasons: hostile environment, robot runs out of on-board battery for instance. This poses a problem of ensuring the coherence of the table with the actual situation of the network or to recover from robots' failure. Here we consider these cases for the optimized algorithm version.

4.1 Lost of communication link and meeting with new robots

There are two situations that might make the information in the table obsolete: robot "meet" new neighbors or it is out of reach of an access robot (caused either by access robot's failure or robot moves out of reach of the other one).

As soon as a robot detects a breakage of any link with an access robot, it will remove all the access paths going through this link in its table. Then it broadcasts a `Link-Broken` message to its neighbors. The message contains id of the sender and of the disconnected access robot. If there are already forwarded paths to delete, robot will select *not-yet-forwarded* paths in its table based on the criteria similar to that in the algorithm 2 (i.e. there are at most two forwarded paths, and these paths must help their neighbors to build as complete as possible their list of access robots). These paths are then sent along with the message.

Any robot receiving a `Link-Broken` message will check in its table and crosses out any access paths through the broken link. The "reserved" paths in the message will be added into its table as well. If there are any forwarded paths to be deleted, robot selects paths in its cache (with the newly added paths) to send with the `Link-Broken` message to its neighbors. Therefore, all robots that might use the broken link will be notified upon, and then update their tables.

Consider the network in figure 2 for example. Suppose that the link $\{2, 5\}$ is broken. Robot 5 detects that it is no longer in touch with robot 2, the path $(1, 2)$ will be removed from its table. Because there is only one forwarded path left, robot will select in its cache and takes out the path $(1, 2, 4)$ to forward along with the `Broken-Link` message. And so on, robots $3, 4$ and 6, upon the reception of this message, will update correctly it table.

When robots meet new neighbors, they will exchange their connectivity table to each other. A modified version of the algorithm 2 will be executed on each robots to detect, store new access paths and notify neighbors about updates.

4.2 Failure recovery

The update mechanism in the section 4.1 ensures that the connectivity table is kept coherent with the actual network configuration in the presence of the mobility. But what will happen if the reference robot fails to work? If the access robot list is complete, the recovering mechanism for such failure is simple as follow: as soon as a robot detects that it does not affiliate to any access robot (the table is empty after an update), it will declare itself as the new reference robot. This declaration is sent along with the `Link-Broken` message. Other robots upon receiving this message will follow the update procedure (section 4.1), and the sub-networks will be formed naturally.

Network partition: in the network in figure 1, the failure of any robot among robots $1, 2$, and/or 5 will disconnect the network into two or more sub-networks. These nodes are identified as critical nodes (c.f. definition 5 in section 5). Suppose, for example, robot 1 failed to work, then the recovering mechanism will re-form the network into two components: the first one consists of robots $2, 3, 4, 7$, and robot $5, 6$ will be grouped into another one.

Merging sub-networks: after the partition, the sub-networks might get close to each other, in that case, these sub-networks should regroup again. In general, we propose that each sub-network has an id which is the id of the reference robot in that group. One possible solution is when two groups "meet", the robots at the boundary will exchange their table with the information on the reference, and the sub-network whose the id is smaller will affiliate to the other one.

5 Robust Network Connectivity

In many applications, mostly in hostile environments, ones wish to ensure that the malfunction of any node or disruption of any link in the network will not cause interuptions in communication of any pair of nodes in the network. Networks having such a connectivity property are said to be robust and fault-tolerant. Obviously, this constraint is much stronger than the "simple" connectivity discussed so far. In this section we point out the relationships between the connectivity awareness with the robustness of network connectivity.

Let $N(v)$ is the set of the neighboring robots of robot v. $A(v) \subseteq N(v)$ and $NA(v) \subseteq N(v)$ are set of the access robots, and set of non-access robots of robot v respectively. We have the following equation:

$$NA(v) = N(v) \setminus A(v). \quad (1)$$

In the network in figure 3, robot 1 is the reference node, we have $A(4) = \{2, 3\}$, $A(2) = \{1\}$, and $A(1) = \emptyset$.

Definition 5 (Critical Node and Critical Link)**.** A node $u \in V$, or a link $e \in E$ of the graph G is critical if its removal from the graph will disconnect the graph into two connected subgraphs or more. Otherwise, it is called non-critical.

Definition 6 (Robust Connected Network)**.** A robot network is said to have a robust connectivity if and only if $\forall u \in V$ (and $\forall e \in E$), u (and e) is non-critical.

Definition 7 (Cycle or Circuit)**.** A closed path without self-intersections starting from and ending at u is called a cycle or circuit, and denoted by $\zeta(u)$.

Proposition 3. For any node $u \in V$ is non-critical if and only if there exists at least one circuit $\zeta(u)$ such that $\forall v \in N(u)$, $v \in \zeta(u)$.

Proof. If such a circuit $\zeta(u)$ does exist, then when we remove u from G there must be one path obtained from $\zeta(u)$ by removing u from $\zeta(u)$. And all the neighboring nodes of u are still connected to each others. Therefore, the node is non-critical.

On the other hand, if u is non-critical in V, then after its removal the graph remains connected. That is we must be able to find a path in form of $(u_{k_1}, v_1, v_2, ..., v_j, u_{k_2}, v_l..., v_h, u_{k_m})$, where $v_i \in V$, and $u_{k_j} \in N(u)$. The circuit $\zeta(u)$ is obtained simply by adding u to the begin and the end of this path. This concludes the proof. □

For referentially connected graph G with the reference robot r, we have the following properties.

Proposition 4. Any non-reference node $u \in V$ is critical if and only if $A(u) \subset N(u)$.

Proof. The proof is quite straightforward and can be deduced directly from the equation 1.

If node u is critical, by definition when remove it from G, there must be some other nodes that are no longer connected to the reference robots. That is, $NA(u) \neq \emptyset$, or $A(u) \subset N(u)$.

If $A(u) \subset N(u)$ and suppose that node u is not critical, then after its removal, the nodes in $NA(u) \neq \emptyset$ are still connected, meaning that these nodes have other access node than u. Thus node u can consider these nodes as its access nodes. This leads to a contradiction with the hypothesis. Therefore, node u is critical. □

For determining if a link is critical, at the first glance, one might suppose that a link connecting two critical nodes is critical. But like we can see from the network in figure 3, this assumption results in wrong determination. With the connectivity awareness, we have the following proposition that crosses out this misconception.

Proposition 5. Any link $e(u, v) \in E$ is critical if and only if $A(u) = \{v\}$ or $A(v) = \{u\}$

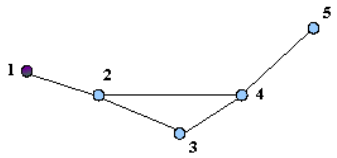


Figure 3: Node 1 is the reference node. 2 and 4 are critical nodes, but the link $\{2, 4\}$ connecting them is not critical.

Proof. If $e(u, v) \in E$ is a critical link, then when the link is broken, either node u or v will be disconnected from the reference node. If the disconnected node is u, then u must have no other access robot than v ($A(u) = \{v\}$). Likewise, if the disconnected node is v, then $A(v) = \{u\}$.

In the case where $A(u) = \{v\}$, then the breakage of this link will separate the nodes connecting to the reference node through u from the sub-component with the reference node. Therefore, $e(u, v)$ is a critical link. Similarly for the case where $A(v) = \{u\}$. □

5.1 Completing the Access List

The propositions presented in section 5 reveals that the verification of whether a network connectivity is robust is a trivial problem and can be done in the most straightforward way once the access list is complete. However the algorithm 2 does not ensure such a requirement. In order to remedy this shortcoming, first we tried many optimizations (c.f. see Appendix 8) to find out the ones that can help to reduce the forward paths, as well as ensure that all the access robots are figured out in the list. Unfortunately the efforts all have failed. But we have an important notice: if the list of access robot on one node is incompleted to be critical, that is because of some of its neighbors did forward only one path. The solution for completing the access list should be solved with such neighbors.

We make use of a mechanism similar to the RREQ (Route Request) and RREP (Route Reply) in the DSR routing protocol [JMB01] to complete the access list. After the reception of the first `New-Access-Path` message, robot will wait for a period of time (defined by `Complete-Waiting-Time-Out` constant). When the time out passed and if the list has not yet been completed, robot will send a message `Complete-Access-Path-Request` to its neighbors who it has received only one `New-Access-Path` message from.

The `Access-Path-Request` message contains:

- *The id of the message*. If a robot receives a message, it will register the identification of this message to not proceed further it within a certain period of time. This trick helps to avoid flooding the network with the same request.
- *A reverse path* used to send back when a robot find such a path. This path begins with id of the original message's issuer, and when it is forwarded by a robot, the sender's id will be incrementally added to the path.

When a robot receives an already treated `Access-Path-Request` message (robot recognizes this by mean of the id of the message and those it has stored), the message will be ignored.

Otherwise, it will look up in its table to find a path satisfying the condition, if such a path is found, robot broadcasts an `Access-Path-Response` message that contains this path. On receiving an `Access-Path-Response` message, a robot will add the complementary path to its table. If its id is in the reverse path, it will remove the id from this path and continue forwarding the message with the updated paths (i.e. the complementary path and the reverse path).

Let us reconsider the network shown in figure 2, after the waiting time is out, robot 2 finds that its neighbors, robot 3 and 4, have just forwarded one `New-Access-Path` message, robot 2 will send out an `Access-Path-Request` message to the two robots. When robot 3 receives this request message, it can not find any path in its table to complete the access list, it broadcasts the message to robot 5. Because robot 5 finds the path $(1, ..., 6)$ that does not go through neither robot 2 nor 3, it broadcasts a new `Access-Path-Response` message to robot 3 and 4. And so forth, the message will reach robot 2 to help it complete the access list. This solution is fruitful with least extra message.

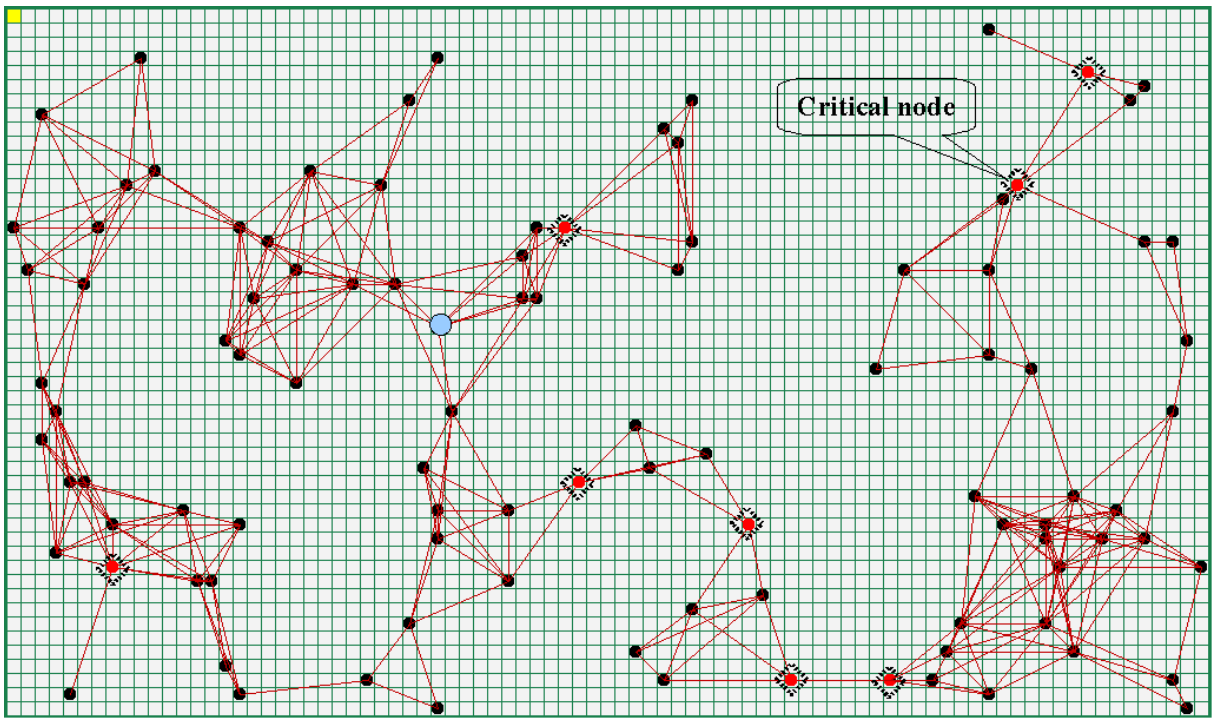


Figure 4: A simulation snapshot of critical nodes detection. The reference robot is the biggest circle. The critical nodes are highlighted by a diamond around. There are 100 nodes in the network, all 8 critical nodes are detected successfully.

5.2 Checking the Robustness of the Network Connectivity

According to the proposition 4, and 5 (section 5), any non-reference node knows that it is critical if the number of its access robots is smaller than that of its neighbors. In addition, if robot has

Table 3: Average total number of messages proceeded per robot with the completion procedure of the access list with the various network configurations.

Statistics Value	Network size				
	15	20	50	100	500
max	6	7	9	9	9
min	2	2	3	3	3
median	3	3	4	4	4

only one access robot, then the link between it with the sole access robot is critical.

The criticalness determination of the reference node is based on the proposition 3. The reference robot after sending the first `New-Access-Path` message, will receive and register all the paths back from its neighbors. From these paths, the reference node will be able to construct a graph and deduce from that whether it is critical or not. Now the detection of critical nodes in network can be carried out simply by waiting for some period of time so that the knowledge converges to the stabilized status, and then apply the above-proven propositions to identify the critical nodes. We carried out series of simulations as described in section 3.4. A snapshot of the simulation is shown in figure 4. The detection of our algorithm is check again with a centralized, global algorithm. Our algorithm detected successfully all critical nodes.

The metric to evaluate the algorithm performance is the communication overhead, i.e. the total number of messages. As observed from table 3, the maximum number of messages sent by a robot is about 10, and the average number is very low. From these simulation, we can figure out that the message complexity[2] to build the complete access list is of $O(kn)$, where k is somewhere between 3 (the best case) and 9 (the worst case). Furthermore, we did not find any clear correlation between the network density and the number of messages sent by robots. This can be explained as robots always forward at most two messages regardless of the number of its neighboring robots.

6 Using Connectivity Awareness in Multi-Robots Distributed Motion Control

The awareness gives a new perceptions on the connectivity for individuals in the network. The maintenance of connectivity can be thus interpreted for individuals in the network as follow: given a reference robot, for preserving the network connectivity, robots need to maintain the communication links with their access robots while performing their tasks. This section presents a preliminary application of integration of the connectivity awareness with distributed motion control in multi robot exploration under the wireless network constraint.

Exploration is one of the main applications in robotics. A popular approach is derived from Yamauchi's work [Yam98]. The basic idea is simple: *in order to gain as much new information*

[2]This number of messages is the total number of messages issued by the algorithm 2 and the access list completion procedure.

as possible about the unkown world, robots in the team need to move to the boundary between known and yet unexplored environment. The boundary is also referred as the frontier; hence the derived approaches are named as *frontier-based robot exploration*. Many works have brought significant extension with a team of robots in order to speed up the exploration process and to decrease the uncertainty in information gained [BMSS05, SYTX06, RB07]. These works mainly focused on proposing efficient collaborating mechanisms so that the overlaps between robots would be minimized. Among others, we choose to modify the algorithm proposed by M. N. Rooker and A. Birk [RB07] for our illustration.

Based on on the Yamauchi's approach, extended with the constraint of wireless network taken into account, Rooker and Birk's algorithm ensures that during the exploration, no robot will loose the connection with the rest of the team. Yet this is a totally centralized approach with an implicit server that collects all the position of the robots in the team. At each iteration, in order to avoid the combinatorial explosion, the server generates a subset of all possible positions for all robots in the team (referred as a *configuration* in their work), then evaluates the generated configurations of the whole system to choose the best one according to the utility function. The same result can be obtained in a distributed fashion using our solution for maintenance of connectivity, and the robots' motion control as well.

6.1 Assumption on Robot Platform and Exploration Algorithm Implementation

As in their work, we model the ground to explore by a 2D occupancy grid, composed of cells. Each cell has one of four possible values: *unknown*, *visited*, *frontier*, and *obstacle*. An unknown cell is the one that has not been visited yet by any robot. As soon as robots position themselves on an obstacle-free cell, it marks this cell as visited, and the neighboring obstacle-free cells will be the frontier cells if their status are still unkown; also, the obstacle cells is sensed by robots when they situate on the cells next to these ones.

During the mission, each robot maintains a map. We assume that robots are capable of localizing itself with respect to its own local map. When robots sense and update the status of yet-unknown cell, they will update this information in their own map and broadcast the update to the teammates (those who are in the same network) as well.

We also assume that the local time on each robot is synchronized at the beginning of the mission, and during the mission as well. The simulations are carried out by exploration time step. An exploration time-step is defined by the period of time for robot to calculate the next cell to move to; and for accomplishing the move. We assume that all the updates with respect to the network topology change are also accomplished within an exploration step. At each iteration (exploration step), the new position is determined as follow: at first robot calculates the closest frontier cell with respect to its present position, then an obstacle-free cell among its neighboring cell will be the new position if this move does not get robot out of the safe-moving zone (to be defined in section 6.2) of its *last* access (cf. the 3 for more details).

Algorithm 3: Robot's Exploration Stepping

Input: A set of frontier cells

Output: The next move, that gets robot closer to the closest frontier cell without breaking the communication link with the last access robot, otherwise, move toward the access robot.

```
1  begin
       // an exploration time-step
2      while frontierCells.isNotEmpty() do
3          target ← the closest frontier cell for robot;
4          calculate the best move toward the target that keep robot in the safe-moving zone
           of at least one acces robot;
5          if such a move was found then
6              move to the new position;
7              broadcast new position to neighboring robots;
8          else
9              move toward the access robots with whom the distance is further than the
               radius of the safe-moving zone r, (but within the communication range R);
10         end
11         if has new exploring information then broadcast update to all teammates;
12     end
13 end
```

6.2 (Sub-)Network formation and partition

In the simulation implementation, we make use of the widely-accepted communication model which is a *unit graph* where the neighborhood-ship is defined based on the Euclidean distance d_e between robots. All robots have the same communication range R, and $e(u, v) \in E \leftrightarrow d_e(u, v) \leq R$. Regarding the maintenance of network connectivity, we define a safe-moving zone that if one robot wishes to stay in touch with its access robot, it must be within this zone (figure 5). This safe-moving zone is determined by a circle centered at robot and has the radius $r < R$. The awareness is now translated into the constraint for selecting a move of a robot: robots should not go out the safe-moving zone of their access robots. The access list on robots is built by the algorithm 2, without the procedure to complete it.

The connectivity constraint introduces an asymmetric dependency between a robot and its access robots: robots depend on their access robots for selecting the next position; as consequent, robots tend to follow their access robots. However, because the calculations are realized in parallel on individual robot, the safe-moving zone is not enough to keep robot always in touch if there is a concurrent situation as shown in figure 6, where robot 2 and robot 3 are access robot of each other, and at the same time, they move out of the communication range of the reference

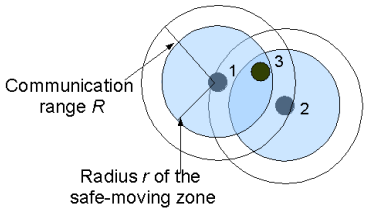


Figure 5: Communication range (the bigger, light-gray circle) and safe-moving zone (the smaller, and darker one). The safe-zone moving of robot 3 is the combination of the safe-zone moving of two its access robots: 1 and robot 2. In order to maintain the connectivity, R_3 should not move out of this zone.

robot 1.

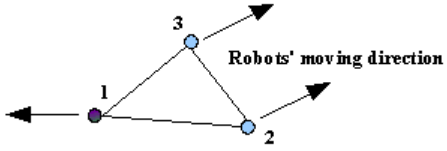


Figure 6: A concurrent situation causing the network partition.

In such a situation, for applications where a permanent connectivity is required, a more sophisticated coordination need to be defined in order to maintain the connectivity. In the work presented here, robots only make use of basic knowledge of the connectivity to keep in touche with each other. When being turned on, a robot starts to look for a network with a reference robot to affiliate to. If the network is not found, it will wait for a random period of time before declaring it self as the reference robot of the new subnetwork. When two sub-networks get close to each other, they will merge into larger one. Robots are randomly deployed on a terrain of extent 45×45 cells. Figure 7 introduces the snapshot of our simulator.

6.3 Simulation Results

First of all we evaluate the performance of the exploration algorithm. The result regarding this evaluation is given in the chart in figure 8. The number of robots in simulations varies from 5 to 10. Although there are many potential improvements that can be added to the coordination, the algorithm scales well: addition of robots to the exploration algorithm reduce linearly the exploration time. We evaluate the performance of the same algorithm in two cases: with and without limiting the communication range. When limiting the communication range, we set $R = 15$, and $r = 12$ in distance[3] unit used in the simulations.

[3] a distance unit is the maximum distance a robot can move within a step.

The simulations are realized to prove the feasibility of using awareness in maintaining the network connectivity; in which, we do not require a permanent connectivity in the group of robots. Furthermore, it is difficult to ensure such a connectivity in the presence of obstacles. Hence, with respect to the evaluation of the connectivity maintenance, we remove all the obstacles and robots are placed close to each other at the beginning of the mission so that they form a network. Then we record the time duration of the network till the first partition takes place. We measured for groups of 5 and 10 robots, with the safe-moving zone radius varies from 7 to 14 distance unit. Each configuration are run repeatedly 20 times. The median value of these runs are shown on the chart in figure 9.

We can draw the conclusion from the chart that the radius of the safe-moving zone has more impact on the network of smaller size. This results are well expected because there are more concurrent processes in larger network leading to the partition of network.

7 Related Works

In this section we briefly review some works that are close to ours in the context of maintaining connectivity in mobile wireless network and the application of these techniques to maintenance of connectivity (distributed motion control under the wireless network constraint) in Multi-Robot Systems.

7.1 Robustness of Connectivity in Wireless Mobile Network

Some previous works have already discussed the issue of critical node detection in MANET. A classical, centralized approach using a DFS (Depth First Search) is presented in [DABS00]. Sheng et al. [SLS06] presented a distributed algorithm, namely DMCC (Detection Algorithm based on Midpoint Coverage Circle). The algorithm first determine whether a node is critical in an area (Midpoint Coverage Circle). Once a node is supposed to be critical, they need to find all global paths between the node and its neighbors to conclude on the global criticalness of the node. Because of the need of all global paths, the algorithm suffers significant communication overhead for the detection. The approach might not scalable. An alternate approach based on the detection of *k-hop* critical node of M. Jorgić et al. [JSHSR04]. Instead of being aware of global network topology, only nodes which are k-hop neighbors exchange the information to rebuild a local view of the connectivity. This work is then extended in [DLNSar] with the procedure to remedy the criticalness once detected. This has the advantage of eliminating the need of global information. However, up to 20% of nodes are falsely declared as global critical due to this compromise.

Similar to work in [JSHSR04, DLNSar], Ahmadi and Stone [AS06a] proposed a distributed algorithm for checking whether a robot network connectivity is robust (referred in their work as *biconnectivity*). This algorithm issues a huge number of message ($O(2n!)$) due to the full exchange of the network topology. In addition, they made a strong assumption on the mobility of the robots: during the execution of checking the biconnectivity, robots should not change the

network connectivity. This work is extended in [AS06b] with the work to move robots in the system in such a way that the network is always "bi-connected".

In short, as compared to these works, advantage of our approach over these works is that: the awareness of the connectivity reveals that the checking of whether a network is tolerant to the connectivity of nodes is a trivial problem and can be done in the most straightforward way with least communication overhead, provided that the reference node is well chosen.

7.2 Connectivity Maintenance in Multi-Robot System

Applying the MANET technologies to the communication of a team of mobile robots is not new in the literature [Win00]. However, to our best knowledge, there has been limited work[4] that consider the awareness of connectivity in a networked robots system as an abstract service that can be incorporated into various applications. Rather, the issue of maintaining the network connectivity is tightly-coupled to the application, hence the solutions are proposed in ad hoc manner. Many works have attempted to take the communication constraints into consideration when planning the motion for robots in MRS. A typical example is the exploration of an unknown environment by a team of robots that communicate in order to collaboratively build up a map.

Vazques and Malcolm [VM04] proposed a solution where robots periodically broadcast their positions and current headings. Based on the information of all the other robots, each robot re-builds the current network topology and tries to maintain the connection with the team. Sheng et al. [SYTX06] proposed a distributed bidding mechanism for robots team exploration. A heuristic utility function based on the *nearness measure* guides each robot to keep it close to the others. In all of these solutions, the global information (all robots' positions) should be propagated to every robot.

Works in [SJK08] are to objective of controlling the motion of a group of exploring robots while maintaining the connectivity with a stationary robot in a walled environment. However, the entire procedure has not been decentralized, but only some parts. The authors in [ZP07b] developed a centralized feedback control framework to drive the agents to configurations away from the undesired space of disconnected networks while avoiding collisions with each others. This work is then decentralized in [ZP07a] but with a communication overhead of $O(n^2)$.

8 Conclusion and Future Work

We presented in this paper a novel approach toward maintaining the connectivity in networked robotics system. The knowledge on the connectivity is built by mean of a distributed algorithm which results in very low communication overhead. The theoretical results of this paper have been confirmed by simulation of various robot network configurations.

Furthermore, in the proposed solution we consider the connectivity awareness as a separated concern that can be reused in various robotic applications with different application-dependent

[4] [ZP07a] introduces a distributed topology control that take into account what they called *secondary objectives*. This notion is very similar to the one we have in mind when we propose the maintenance of connectivity as a transverse concern.

strategies for connectivity maintenance as illustrated in two applications of checking the robustness and of controlling the robots motion in this paper. As compared to existing works, our solution is much more efficient in term of communication overhead. For checking the robustness of the connectivity, our algorithm requires $O(kn)$ messages (k is somewhere between 3 and 9) while Ahmadi and Stone's solution [AS06a, AS06b] requires $O(2n!)$ messages[5]; or to build the knowledge on network connectivity, our algorithm issues $O(2n)$ messages, whilst Zavlanos and Pappas's [ZP07a] work, very close to ours in purpose, has message complexity of $O(n^2)$.

Regarding future work, our starting point is to investigate further on a protocol for more sophisticated coordinations between robots in environment with the presence of obstacles using the awareness of the connectivity. The problem of selecting the reference robot dynamically would be also worth pursuing.

Appendix A: Message Complexity of the Algorithm 1

Proposition 6. In the worst case (i.e. a complete graph), the space complexity for saving a connectivity table message complexity of the algorithm are both $O(n!)$.

Proof. When the graph is complete, i.e. all the robots in the network are all neighbors of each others; thus, any robot has $n-1$ neighboring robots. For robot R_i, the message issued by the reference robot reaches (and will be added to the depending table) it through all non-acyclic paths. In a complete graph, there are such $(n-2)![1+\frac{1}{1!}+\frac{1}{2!}+...+\frac{1}{(n-2)!}]$ paths. The space complexity for saving the connectivity table is thus $O(n!)$

We will count the number of messages. First, the reference robot first sends out the message to $n-1$ other robots. The next step, all $n-1$ robots will forward the message to all their neighbors, hence there are $n-1$ messages. A message will stop being forwarded when it gets back to the node that has sent it out before. Thus, at k^{th} step, there are $n-k-1$ robots that still send out message. As consequent, we have totally $1+(n-1)!$ messages or in other words, the messages complexity of the algorithm is of $O(n!)$. □

Appendix B: Optimized Messages Forwarding with Disjoint Paths

Another way to reduce the number of messages traveling in the network based on the conception of disjoint path (c.f. definition 8).

Definition 8 (Internally disjoint paths)**.** Two different paths $p_1(u,v)$ and $p_2(u,v)$ are internally disjoint if they have no vertices in common except u and v.

Proposition 7. In the algorithm 4, the message complexity of the algorithm is $O(2n\bar{d})$ where $\bar{d}$ is the average number of neighbors.

[5]Besides, our solution applies even while robots are moving, as opposite to Ahmadi and Stone's work that applies only if the network topology is freezed.

Algorithm 4: Algorithm to be executed upon the reception of a `New-Access-Path` message – filter with disjoint paths

Input: The `New-Access-Path` Message M
Output: The connectivity table T of the robot is updated

```
1 begin
2     p ← the access path in M;
3     if this.id() ∉ p then
4         add the path p to the connectivity table T;
5         if the path is disjoint with all the paths in the table T then
6             update and forward the message to all neighbors;
7         end
8     end
9 end
```

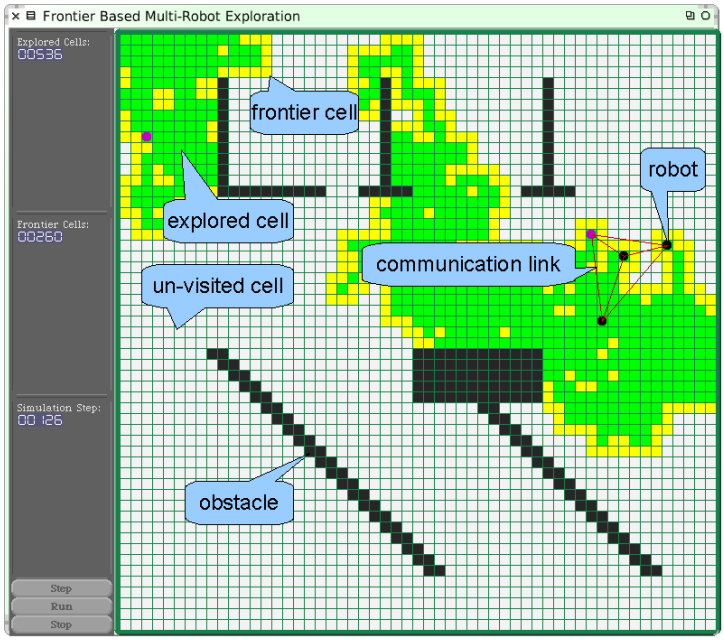
Proof. We first consider the worst case when the graph is complete, i.e. all the robots in the network are all neighbors of each others; thus, any robot has $n-1$ neighboring robots. For any robot R_i, the message issued by the reference robot reaches R_i through all possible loop-free paths. However, robots do not forward all the paths, but only those that are internally disjoint (c.f. definition 8 in 2) to each other. To count the number of access paths, we classify them according to the length. Let denote C_l is the number of the paths having length l. All the paths of length 1 and 2 are by definition internally disjoint. They will all be added to the table. $C_1 = 1$, and $C_2 = n-2$. For counting $C_k, 3 \leq k \leq n-1$, we notice that since they are all internally disjoint with each other, a neighboring robot of R_i can not appears more than once in all the paths whose length is greater than or equal to 3. Thus, $\sum_{k=3}^{n-1} C_k \leq n-2$, where $n-2$ is the number of neighbors excluding the reference robot. Totally, we have $1 + (n-2) + (n-2) = 2n-3$ possible access paths sent by a robot.

During the initialization phase, the reference robot first sends out the `New-Access-Path` message to $n-1$ other robots. Since then the reference robot will not send any more message. For the other $n-1$ robots, each robot sends at most $2n-3$ messages, the total number of message in the network is $(2n-3)(n-1)+1$. So the message complexity is $O(2n^2)$.

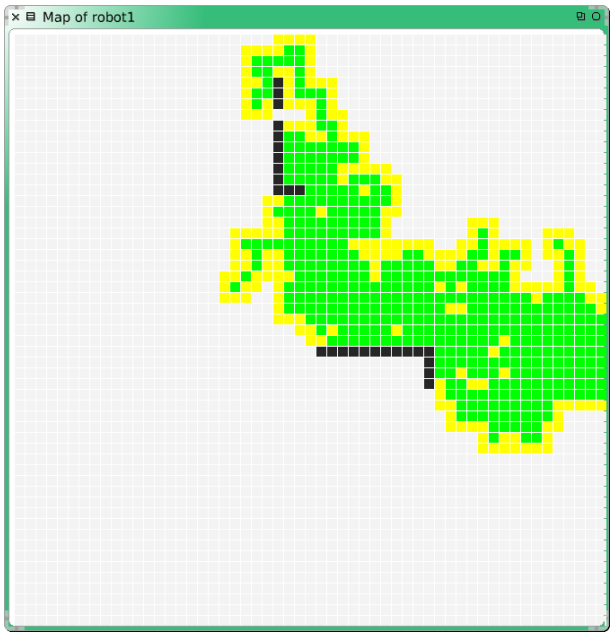
Note that the complexity above is for the worst case. In general for the networks with referentially connected graphs having the average degree $\bar{d}(d << n)$, with arguments similar to the above ones, the message complexity is $O(2n\bar{d})$. □

Acknowledgment

The authors would like to thank Arnaud Doniec, from the Ecole des Mines de Douai for his kindly help during the preparation of this paper.



(a) The Simulation main screen.



(b) A sample of robots' local map.

Figure 7: A Simulation snapshot of frontier-based multi-robot exploration.

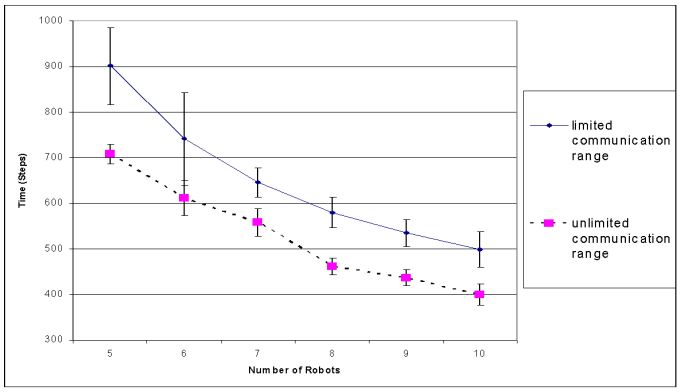


Figure 8: Performance of the exploration algorithm with and without limiting the communication range.

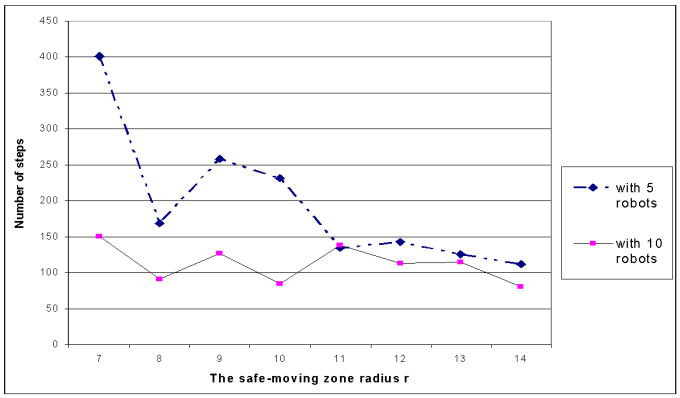


Figure 9: Duration of robot network before the first partition.

References

[AS06a] Mazda Ahmadi and Peter Stone. A distributed biconnectivity check. In *proceeding of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS'06)*, 2006.

[AS06b] Mazda Ahmadi and Peter Stone. Keeping in touch: Maintaining biconnected structure by homogeneous robots. In *proceeding of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

[BA94] T. R. Balch and R. C. Arkin. Communication in reactive multiagent robotic system. *Autonomous Robots*, 1(1):1–25, 1994.

[BMSS05] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank SchneiderâĂă. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[DABS00] M. Duque-Anton, F. Bruyaux, and P. Semal. Measuring the survivability of a network: connectivity and rest-connectivity. *Transaction of Telecommunications*, 11(2):149–159, 2000.

[DLNSar] Shantanu Das, Hai Liu, Amiya Nayak, and Ivan Stojmenovic. A localized algorithms for bi-connectivity of connected mobile robots. *Telecommunication Systems*, 2008 (to appear).

[JMB01] David B. Johnson, David A. Maltz, and Josh Broch. *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*, chapter 5. Addison-Wesley, 2001.

[JSHSR04] Milenko Jorgic, Ivan Stojmenovic, Michael Hauspie, and David Simplot-Ryl. Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks. In *Proceeding of The Third Annual Mediterranean Ad Hoc Networking Workshop*, 2004.

[Mac91] B. MacLennan. Synthethic ethology: An approach to the study of communication. In *Proceeding of the Second interdisciplinary workshop on synthesis and simulation of living systems*, pages 631–658, 1991.

[Per01] C. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2001.

[RB07] Martijn N. Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435–445, 2007.

[SJK08] Ethan Stump, Ali Jadbabaie, and Vijay Kumar. Connectivity management in mobile robot teams. In *Proceeding of the 2008 IEEE International Conference on Robotics and Automation*, pages 1525–1530, 2008.

[SLS06] Min Sheng, Jiandong Li, and Yan Shi. Critical nodes detection in mobile ad hoc network. In *20th International Conference on Advanced Information Networking and Applications (AINA'06)*, volume 2, pages 336–340, 2006.

[SYTX06] Weihua Sheng, Qingyan Yang, Jindong Tan, and Ning Xi. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54:945–955, 2006.

[VM04] Jose Vazques and Chris Malcolm. Distributed multirobot exploration maintaining a mobile network. In *Proceedings of 2nd International IEEE Conference Intelligent Systems, 2004.*, volume 3, pages 113–118, 2004.

[Win00] A. F. T. Winfield. Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. *Autonomous Robotic Systems*, pages 273–282, 2000.

[Yam98] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceeding of the second International Conference on Autonomous Agents (Agent'98)*, 1998.

[ZP07a] Michael M. Zavlanos and George J. Pappas. Distributed connectivity control of mobile network. In *Proceeding of the 46th IEEE Conference on Decision and Control*, pages 3591–3596, 2007.

[ZP07b] Michael M. Zavlanos and George J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4), 2007.