Fault-Tolerance Through k-Connectivity

Alejandro Cornejo
acornejo@csail.mit.edu
MIT CSAIL

Nancy Lynch
lynch@csail.mit.edu
MIT CSAIL

***Abstract*—We consider a system composed of autonomous mobile robots that communicate by exchanging messages in a wireless ad-hoc network. In this setting, the failure of a single robot might result in the disconnection of the communication network. We are concerned with the problem of maintaining a fault-tolerant connected network while allowing robots to perform other tasks.**

Specifically we describe two local distributed algorithms to determine if a graph is k-connected. We then describe how these algorithms could be used to extend the connectivity maintenance algorithm of [2, 3] to maintain a k-connected network while allowing robots to perform other tasks.

I. Introduction

One of the main concerns in multi-robot systems is fault-tolerance. The expected number of robot failures grows linearly with the population of a system. Moreover, multi-robot systems typically lack a centralized communication infrastructure and rely on ad-hoc networks for communication. In this setting, coordination is possible only if the communication graph remains connected. Therefore, to design fault-tolerant multi-robot systems, it is paramount that the failure of a single robot does not compromise the correctness of the whole system. These issues are only aggravated by the mobility of the robots, since this means that the communication topology is changing constantly.

The connectivity of a graph is a good estimate of the fault-tolerance of the network, since higher connectivity means more robots can fail without disrupting the communication among the rest of the robots. More precisely, a graph with $n > k+1$ nodes is *k-connected* if any set of $k-1$ vertices can be removed without disconnecting the graph. By convention complete graph on n nodes is $(n-1)$-connected. The connectivity of a graph is the maximum k for which it is k-connected.

Designing algorithms to efficiently perform tasks using multi-robot systems is difficult even without the added burden of explicitly dealing with problems such as graph connectivity and fault-tolerance. To avoid this added complexity researchers often ignore the aspect of graph connectivity. Instead they design algorithms that control the motion of the robots but that lack any mechanisms to prevent graph disconnection. This includes work on flocking [10, 6], pattern formation [5], and leader following [1], among others.

Our aim is to design a separate algorithm that explicitly deals with the problem of maintaining k-connectivity, while allowing the robots to perform other tasks. For such an algorithm to be useful in practice, it should not make any restrictive assumptions on the tasks the robots are executing. Moreover, to scale to systems with large populations of robots, the algorithm should rely only on local knowledge. We envision that roboticists, aided by such an algorithm, could design fault-tolerant cooperative multi-robot systems for a variety of tasks with little added effort.

In previous work [2, 3] we addressed the problem of maintaining connectivity ($k = 1$) for robot swarms while performing arbitrary tasks. Specifically, we described a distributed algorithm that modifies an existing motion plan to ensure connectivity. The algorithm uses only local information, is stateless, does not require a fixed set of neighbors and does not make any assumptions on the current or goal configurations. Moreover, the algorithm is *robust* to the robots' speed changes; if robots travel any fraction of the trajectory (perhaps none) at any speed, connectivity is preserved. The *progress* of the algorithm is defined as the total distance traveled by all robots (summing over all the robots) towards their intended destination. Let d be the minimal distance each robot intends to move and let r be the communication radius. Assuming that the target configuration of the robots is connected and the motion does not require breaking any cycles, we proved that the algorithm guarantees the progress is at least $\min(d, r)$. Furthermore, we exhibited a class of configurations where no local algorithm can do better than this bound, and hence under these conditions the bound is tight and the algorithm is asymptotically optimal. Finally we proved that all robots get ε-close to their target within $\mathcal{O}(D_0/r + n^2/\varepsilon)$ rounds where D_0 is the total initial distance to the targets and n is the number of robots [3].

In this paper we describe two distributed algorithms that rely only on local knowledge and can be used to determine if a graph is k-connected. We describe how to use these algorithms to extend our work in [2, 3] to maintain k-connectivity.

II. Related Work

Most of the previous work on k-connectivity is in the field of topology control. Jorgic et al. [8] report the experimental results of three different distributed algorithms to detect k-connectivity on random geometric graphs, but lack any formal guarantees. Czumaj and Zhao [4] presented a greedy centralized algorithm to construct a k-connected t-spanner with runtime $\tilde{O}(nk)$. Thurimella [11] described a distributed algorithm to identify sparse k-connected subgraphs. However, their algorithms require global knowledge, and in fact run

This work was supported in part by AFOSR (Award Number FA9550-08-1-0159) and NSF (Award Numbers CNS-0715397 and CCF-0726514).

in $\tilde{O}(diam(G) + \sqrt{n})$ time. Jia et al. [7] considered the problem of minimizing the power assignment while preserving k-connectivity. In particular they described a centralized algorithm to approximate the minimum power assignment under different assumptions on k. The work of Li and Hou [9] is closest to the work described in this paper. They describe a local distributed algorithm to construct a k-connected spanner. However, their results depend on the assumption that the original graph is k-connected and there is no mechanism to detect when this is not the case.

III. Model and Definitions

In this section we describe our communication model along with our assumptions about the world and the capabilities of the multi-robot system.

Each robot has a unique id and is capable of determining the relative positions of neighboring robots. Except for the state variable where the unique id is stored, the start state of each robot is assumed to be identical. Furthermore all robots run the same deterministic algorithm. We assume a synchronous network model; the system progresses in synchronous lock-step rounds. At every round each robot can send a message to its neighbors, receive messages from its neighbors and perform local computation. Under this model, after t rounds of communication a node v can learn about other nodes which are t hops away, but no more. An algorithm is *local* if it runs for a constant (independent of n or the diameter) number of rounds.

Let V represent a set of robots, where each robot $v \in V$ is associated to a point in Euclidean space $x_v \in \mathbb{R}^2$. We use $\|a - b\|$ to denote the Euclidean distance between points a and b. We model the topology of the communication network as unit disk graph $G = (V, E)$. Concretely, G is an undirected graph with an edge between two robots if and only if their Euclidean distance is less than one ($E := \{(u, v) | \, \|x_u - x_v\| \leq 1\}$). We define the weight of an edge (u, v) as the Euclidean distance between its endpoints, $w(u, v) = \|x_u - x_v\|$. Unique ids can be used to consistently break ties between edge weights, so without loss of generality we assume distinct edge weights. The unit disk graph assumption on the communication graph is required not only for some k-connectivity tests, but also by the algorithm to preserve k-connectivity described in Section V.

We define $N(u) = \{v \mid (u, v) \in E\}$ as the set of nodes that are neighbors of u in G, and $N[u] = N(u) \cup \{u\}$ as the closed neighbors of u. For a positive integer t we denote with $N^t[u]$ the closed t-neighbors of u, the set of nodes reachable by paths starting at u and of length at most t. Let $G(u)$ be the graph induced by the closed neighbors of u, and $G^t(u)$ be the graph induced by the closed t-neighbors of u.

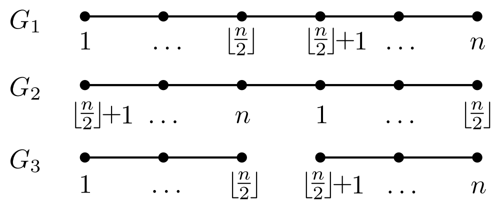
A *vertex cut* C of a connected graph G is a set of vertices whose removal renders G disconnected. The *size* of a vertex cut C is the number of vertices $|C|$. A vertex cut is said to be a *minimum vertex cut* if it is a vertex cut of smallest size. The connectivity of a graph G, denoted by $\kappa(G)$, is the size of a smallest vertex cut of G. A complete graph on n vertices has no cuts at all, but by convention its connectivity is $n - 1$.

We are interested in designing local algorithms to test and preserve k-connectivity. Since connectivity, and more generally k-connectivity, are global graph properties we first define formally what we mean by locally testing a global property.

A *test* is an algorithm $\mathcal{A}$ that produces an accept or reject output. We use $\mathcal{A}_v$ to denote the output of running algorithm $\mathcal{A}$ at node v. Algorithm $\mathcal{A}$ is a *safe* test for a global property $\mathbf{P}$ of a graph, if when algorithm $\mathcal{A}$ is run in a graph and all nodes accept, the graph satisfies $\mathbf{P}$. (Formally, $(\forall v \in V \quad \mathcal{A}_v)$ accepts $\Rightarrow \mathbf{P}$.) This allows for false negatives, since even if some node rejects, the property may still hold, but does not allow false positives. We say test $\mathcal{A}$ is *more accurate* than test $\mathcal{B}$ if the graphs where $\mathcal{A}$ produces false negatives are a strict subset of the graphs where $\mathcal{B}$ produces false negatives. Test $\mathcal{A}$ is *accurate* if when run in a graph that satisfies $\mathbf{P}$ all nodes accept. (Formally, $(\forall v \in V \quad \mathcal{A}_v$ accepts$) \Leftarrow \mathbf{P}$.)

IV. Testing k-Connectivity

We start by showing that we cannot achieve simultaneously safety and accuracy when considering local tests for connectivity.



G_1
1 ... $\lfloor\frac{n}{2}\rfloor$ $\lfloor\frac{n}{2}\rfloor$+1 ... n
G_2
$\lfloor\frac{n}{2}\rfloor$+1 ... n 1 ... $\lfloor\frac{n}{2}\rfloor$
G_3
1 ... $\lfloor\frac{n}{2}\rfloor$ $\lfloor\frac{n}{2}\rfloor$+1 ... n

Fig. 1. Vertices are numbered from left to right. The label of each vertex is displayed directly below it.

Theorem 1. *There does not exist an accurate safe local test for graph connectivity.*

Proof: Consider the graphs G_1, G_2 and G_3 over the vertex set $V = \{1, \ldots, n\}$. Every graph G_i has a labeling function $\ell_i : V \to N$, where $N = \{1, \ldots, n\}$ is the space of unique ids.

Graphs G_1 and G_2 are line graphs, where for every vertex $i \in [1, \ldots, n-1]$ there exists an edge $(i, i+1)$. Graph G_3 is identical to G_1 but without the edge $(\lfloor\frac{n}{2}\rfloor, \lfloor\frac{n}{2}\rfloor + 1)$, and therefore it is disconnected. Both G_1 and G_3 are labeled with the identity function, $\ell_1(i) = \ell_3(i) = i$. For G_2 we define the labeling function $\ell_2(i) = \begin{cases} i + \lfloor\frac{n}{2}\rfloor & i \leq n - \lfloor\frac{n}{2}\rfloor, \\ i - \lfloor\frac{n}{2}\rfloor & \text{otherwise.} \end{cases}$

Fix any local test $\mathcal{A}$ that runs for t rounds. By the definition of local we can assume $t < \lfloor\frac{n}{4}\rfloor$. Each node u observes as $G^t(u)$ a line graph with at most $2t + 1$ vertices. We refer to this line graph with the associated labels as a t-labeled-neighborhood, which can be represented by an ordered list of ids.

We proceed to describe the t-labeled-neighborhood observed by a node with unique id i in G_3.

1) If $i \in [1, \lfloor \frac{n}{4} \rfloor]$ its t-labeled-neighborhood is: $[\max(1, i-t), \ldots, i, \ldots, i+t]$.
2) If $i \in [\lfloor \frac{n}{4} \rfloor + 1, \lfloor \frac{n}{2} \rfloor]$ its t-labeled-neighborhood is: $[i-t, \ldots, i, \ldots, \min(i+t, \lfloor \frac{n}{2} \rfloor)]$.
3) If $i \in [\lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{3n}{4} \rfloor]$ its t-labeled-neighborhood is: $[\max(\lfloor \frac{n}{2} \rfloor + 1, i-t), \ldots, i, \ldots, i+t]$.
4) If $i \in [\lfloor \frac{3n}{4} \rfloor + 1, n]$ its t-labeled-neighborhood is: $[i-t, \ldots, i, \ldots, \min(i+t, n)]$.

In cases 1 and 3 a node with unique id i observes the same t-labeled-neighborhood in G_3 as it would on G_1. Similarly in cases 2 and 4 a node with unique id i observes the same t-labeled-neighborhood in G_3 as it would on G_2.

If $\mathcal{A}$ is accurate then all nodes must accept when running in G_1 or G_2. However, if $\mathcal{A}$ is safe at least one node must reject when running in G_3. Since two nodes with the same unique id that see the same t-labeled-neighborhood must have the same output, it follows that it is impossible for $\mathcal{A}$ to be both accurate and safe. ■

Since we can't expect a local algorithm to be both accurate and safe, we focus on safety at the expense of accuracy. Specifically, under the assumption that the graphs are connected, we describe two safe local tests for k-connectivity for $k \geq 2$. In subsection A we describe a very natural test for k-connectivity, and in subsection B we describe a local test with lower accuracy but better suited to preserving connectivity.

A. A Natural Local Test for k-Connectivity

Perhaps the most natural algorithm is for each node to test whether its local neighborhood is k-connected. Consider the local test $\mathcal{K}(k,t)$ that runs for t communication rounds and collects its t hop neighborhood information. When run at node u it accepts if the subgraph $G^t(u)$ is k-connected and rejects otherwise.

Algorithm 1 $\mathcal{K}(k,t)$ running at node u

```
1: Run for $t$ rounds and construct $G^t(u)$
2: if $\kappa(G^t(u)) \geq k$ then
3:     accept
4: else
5:     reject
6: end if
```

The next theorem proves that $\mathcal{K}(k,t)$ is indeed a safe test for k-connectivity.

Theorem 2. *When running in connected graphs, $\mathcal{K}(k,t)$ is a safe local test for k-connectivity.*

Proof: Suppose by contradiction that $\mathcal{K}(k,t)$ accepts at every node but G is not k-connected. For $\mathcal{K}(k,t)$ to accept at node u, then $G^t(u)$ must have at least $k+1$ vertices, hence $|V| \geq k+1$. Therefore there exists a vertex cut, of size at least 1 (since G is connected) and at most $k-1$ (since G is not k-connected), such that removing the vertices of the vertex cut disconnects G.

In particular let C denote a minimum vertex cut, and let P and Q be two connected components produced by removing all vertices in C. Fix any vertex $u \in C$, it follows that: (i) $N^t[u]$ contains at least $k+1$ vertices. Otherwise the induced subgraph $G^t(u)$ would not be k-connected and $\mathcal{K}(k,t)_u$ would reject. (ii) There exists a pair of vertices $v, w \in N(u)$ such that $v \in P$ and $w \in Q$. Otherwise u doesn't bridge these two components, and the set $C' = C \backslash \{u\}$ would describe a vertex cut of smaller size which separates P and Q.

Therefore the set $U = N^t[u] \setminus \{v, w\}$ is at least of size $|U| \geq k-1$ and (since $G^t(u)$ is k-connected) removing any subset of U of size at most $k-1$ leaves a path from v to w in $G^t(u)$.

However by assumption removing the set $C \subseteq V$ of size at most $k-1$ disconnects v from w in G. Thus, removing the subset of U defined as $U \cap C \subseteq U$ disconnects v from w in the graph $G^t(u)$. Moreover since $|C| \leq k-1$ then $|U \cap C| \leq k-1$ – a contradiction. ■

It's not immediate how to improve the accuracy of $\mathcal{K}(k,t)$. Fix a k-connected graph $G = (V, E)$ and a node $u \in V$ such that $\mathcal{K}(k,t)_u$ rejects, and hence $G^t(u)$ is not k-connected. Observe that regardless of the algorithm, after t rounds node u cannot distinguish between G and $G^t(u)$, where G is k-connected and $G^t(u)$ is not k-connected.

It is tempting to go further and argue that since safe local tests cannot have false positives, then if $\mathcal{K}(k,t)_u$ rejects, it must be that any other safe local test that runs for t rounds must also reject at u; which would imply $\mathcal{K}(k,t)$ is the optimal local test in terms of accuracy. However, this is not the case, since a safe test requires only that some node rejects. Next subsection describes a safe local tester that is less accurate, but that can be used directly to preserve k-connectivity using the techniques presented in [3].

B. k-Connectivity With Small Edges

Intuitively, one expects that if all the robots move closer to each other (hence decreasing the weight of the edges) the connectivity of the graph should increase. This is exploited by the local tester $\mathcal{UDG}(k,t)$, which relies heavily on the unit disk graph assumption. Informally speaking, this tester will accept if the graph observed after t rounds has a connected spanning subgraph using only "small" weights.

Specifically, when run at node u the tester $\mathcal{UDG}(k,t)$ runs for t communication rounds and constructs $G^t(u)$. The tester then prepares the graph $G'(u)$ by removing all edges of $G^t(u)$ of weight larger than $\frac{1}{k}$. Finally, the tester accepts if $G^t(u)$ contains at least $k+1$ vertices and $G'(u)$ is connected, rejecting otherwise.

Theorem 3. *When running in connected graphs, $\mathcal{G}(k,t)$ is a safe local test for k-connectivity.*

Proof: By Theorem 2 it suffices to show that for every vertex $u \in V$ if $\mathcal{UDG}(k,t)_u$ accepts then $\mathcal{K}(k,t)_u$ accepts.

Algorithm 2 $\mathcal{UDG}(k,t)$ running at node u

```
1: Run for $t$ rounds and construct $G^t(u) = (N^t[u], E[u])$.
2: Let $G'(u) = (N^t[u], E'[u])$
3:   where $E'[u] = \{(u,v) \mid (u,v) \in E[u] \wedge \|x_u - x_v\| \leq \frac{1}{k}\}$
4: if $|N^t[u]| \geq k+1$ and $\kappa(G'(u)) \geq 1$ then
5:     accept
6: else
7:     reject
8: end if
```

Fix a node u where $\mathcal{UDG}(k,t)$ accepts. If $G^t(u)$ is a clique then $\mathcal{K}(k,t)$ accepts and we are done, so suppose otherwise. Let C denote a minimum vertex cut of $G^t(u)$, and let P and Q be two connected components produced by the cut C.

Since $\mathcal{UDG}(k,t)_u$ accepted, then for any pair of vertices $p \in P$ and $q \in Q$ there exists a path $p \rightsquigarrow q$ from p to q in $G'(u)$. We use the vertices of C to define a *gap* in $p \rightsquigarrow q$, as a maximal set of contiguous vertices in $p \rightsquigarrow q$ that belong to C. For each gap g let $g.first$ and $g.last$ be the vertices in the path immediately before and after the gap.

By construction all edges weights in $p \rightsquigarrow q$ are less than $\frac{1}{k}$, hence for any gap g, the Euclidean distance between $g.first$ and $g.last$ is bounded by $(|g|+1)/k$. For every gap g it holds that $|g| \leq |C|$. Hence if $|C| \leq k-1$, the distance between the $g.first$ and $g.last$ is less than $k/k = 1$, and by the unit disk graph assumption there exists an edge $(g.first, g.last)$ in $G^t(u)$ which bridges the gap. Therefore, for C to be a cut, it must be of size at least k, which implies $G^t(u)$ is k-connected. ■

In the process of proving Theorem 3 we showed that the graphs which are detected as k-connected by $\mathcal{UDG}(k,t)$ are a subset of those detected by $\mathcal{K}(k,t)$. It is not difficult to construct graphs where $\mathcal{K}(k,t)$ detects the graph as k-connected while $\mathcal{UDG}(k,t)$ fails (i.e. a clique with $k+1$ nodes using only "large" edges). Therefore $\mathcal{K}(k,t)$ is more accurate than $\mathcal{UDG}(k,t)$. However, as described in the next section, $\mathcal{UDG}(k,t)$ can be used directly to preserve k-connectivity.

V. Preserving k-Connectivity

Starting with a connected graph that is detected as k-connected by $\mathcal{UDG}(k,t)$ this section describes how the proof of Theorem 3 can be used to design an algorithm that preserves global k-connectivity of a graph, while allowing robots to perform other tasks.

The connectivity maintenance algorithm described in [3] preserves simple connectivity while allowing the robots to perform other tasks. It has three phases: a collection phase, a proposal phase and an adjustment phase. In the collection phase, each robot learns its set of neighbors and their relative positions. In the proposal phase, the neighbors are filtered and a local optimization problem is solved to find a new trajectory. Finally in the adjustment phase, the proposed trajectory is broadcast and adjusted to guarantee connectivity. For more details we refer the reader to [3].

If the communication radius of the robots is r, let $r' = r/k$. Suppose we run the connectivity service with this smaller communication radius, dropping all messages from nodes which farther than r'. The guarantees of the service will ensure that the agents make progress towards their target, while at the same time maintaining a connected communication graph with respect to the communication radius r'.

Due to lack of space, we omit the details, but using the same arguments of Theorem 3 it is possible to show that the resulting graph is indeed k-connected.

VI. Future Work

In future work we will consider different local tests, and examine the trade off between safety and accuracy in more detail. We also plan to address the problem of controlling the motion of the robots to repair k-connectivity due to failures.

References

[1] S. Carpin and L. E. Parker. Cooperative Leader Following in a Distributed Multi-Robot System. *ICRA*, 2002.

[2] A. Cornejo and N. Lynch. Connectivity Service for Mobile Ad-Hoc Networks. *Spatial Computing Workshop*, 2008.

[3] A. Cornejo, F. Kuhn, R. Ley-Wild, and N. Lynch. Keeping mobile robot swarms connected. September 23-25 2009.

[4] A. Czumaj and H. Zhao. Fault-tolerant geometric spanners. *Discrete and Computational Geometry*, 32(2):207–230, 2004.

[5] R. Fierro and A.K. Das. A modular architecture for formation control. *Robot Motion and Control*, 2002.

[6] A.T. Hayes and P. Dormiani-Tabatabaei. Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. In *ICRA*, 2002.

[7] X. Jia, D. Kim, S. Makki, P.J. Wan, and C.W. Yi. Power assignment for k-connectivity in wireless ad hoc networks. *Journal of Combinatorial Optimization*, 9(2): 213–222, 2005.

[8] M. Jorgic, N. Goel, K. Kalaichevan, A. Nayak, and I. Stojmenovic. Localized detection of k-connectivity in wireless ad hoc, actuator and sensor networks. *Proc. 16th ICCCN*, 2007.

[9] N. Li and J.C. Hou. FLSS: a fault-tolerant topology control algorithm for wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 275–286. ACM New York, NY, USA, 2004.

[10] A. Regmi, R. Sandoval, R. Byrne, H. Tanner, and CT Abdallah. Experimental Implementation of Flocking Algorithms in Wheeled Mobile Robots. In *American Control Conference, 2005. Proceedings of the 2005*, pages 4917–4922, 2005.

[11] R. Thurimella. Sub-linear distributed algorithms for sparse certificates and biconnected components. In *PODC*, pages 28–37. ACM New York, NY, USA, 1995.