2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops

Connectivity Service for Mobile Ad-Hoc Networks

Alejandro Cornejo and Nancy Lynch
MIT Computer Science and Artificial Intelligence Laboratory
{acornejo,lynch}@csail.mit.edu

***Abstract*—We present a distributed connectivity service that allows agents in a mobile ad-hoc network to move while preserving connectivity. This allows unmodified motion planning algorithms to control the trajectories of each robot independently, these trajectories are fed to the service which modifies them as little as possible to ensuring global connectivity. Since we require only short term targets the service can be used with online motion planning where the complete trajectory is not known a priori. For most motions the algorithm requires only local knowledge of the graph, and therefore scales up with the number of agents.**

I. Introduction

In mobile ad-hoc networks the movement of the agents translates to changes in the underlying communication topology. As an algorithm designer for these platforms we seek two contradicting properties, on one hand we would like to coordinate the motion of the agents to guarantee the connectivity of the communication network, and on the other we would like each agent to perform its individual task while interfering as little as possible with the other agents.

In this paper we propose the use of a connectivity service which mediates between the individual tasks of each agent while guaranteeing the connectivity of the underlying communication graph. The use of this service allows more flexibility in the design of controllers and algorithms for mobile agents, since for the most part the agents can be controlled independently and rely on the connectivity service to modify the trajectories as needed to guarantee connectivity.

In the control theory community Schuresco et al. [1] present an algorithm to transform the connection topology between two trees. Zavlanos et al. [2] developed a centralized method to preserve connectivity by solving a constrained optimization problem. Sevla et al. [3] describe an algorithm to maintain connectivity for second order agents which requires preserving edges. Zavlanos et al. [4] propose using potential fields to steer the agents and describe how to follow a leader using this method. Similar but unrelated work has been done in the field of topology control, for example in [5],[6].

In this paper we present a distributed connectivity service that does not make assumptions over the current and target configurations. In particular the target configuration might be disconnected, and the service guarantees connectivity while trying to get each agent as close as possible to their target. Furthermore, the connectivity service does not require long term knowledge of the target and is in fact memoryless. We define robustness to velocity changes of each agent and prove the proposed connectivity service is robust to any number of these changes. To improve progress our framework allows different filtering methods to be used and we propose filtering methods which require only local information and perform as well as global filtering in a lot of practical cases.

Section II introduces graph theory notation and define trajectories and motion patterns along with some of their properties. In section III we describe an abstract connectivity service and state its properties. Section IV presents the intersecting disks connectivity service, describes some variants and proves their correctness. Section V discusses progress of the intersecting disks connectivity service for connected motions. Section VI contains simulation results with different parameters. Section VII concludes the paper and describes future research directions.

II. Preliminaries

We assume agents have access to some positioning device such as GPS, but it is enough to have the relative position of one hop neighbors. Furthermore, we suppose the position of the agents is restricted to a plane, but the results presented easily extend to general three dimensional movement.

Let $V = \{1, \ldots, n\}$ be the set of agents, for $i \in V$, $x_i(t) \in \mathbb{R}^2$ denotes the position of agent i at time t in 2-dimensional space, which is considered an 2×1 column vector. We use bold capital letters to refer the configuration of the whole system, $\mathbf{X}(t) = [x_1(t), \ldots, x_n(t)]$ is a $2 \times n$ matrix which describes the positions of the n agents at time t. Although we deal with continuos motion, we omit t when discussing positions fixed at one instant.

The closed disk of radius r centered at x_i is defined as $D_r(x_i) = \left\{p \in \mathbb{R}^2 \mid \|p - x_i\| \leq r\right\}$. We concern ourselves with networks where each agent has the same communication radius r and there is a bidirectional link between i and j at configuration $\mathbf{X}$ if $\|x_i - x_j\| \leq r$. Therefore the communication graph between agents is the unit disk graph parametrized with the configuration of the system.

Definition 1. The *communication graph* for configuration $\mathbf{X}$ is defined as $UDG(\mathbf{X}) = \langle V, E(\mathbf{X})\rangle$, where $E(\mathbf{X}) = \{(i,j) \mid \forall i \neq j \text{ s.t. } \|x_i - x_j\| \leq r\}$ for some fixed communication radius r.

A configuration $\mathbf{X}$ is connected if $UDG(\mathbf{X})$ is connected.

A path P from v_1 to v_n is a graph defined by a vertex set $V = \{v_1, \ldots, v_n\}$, with an edge set $E = \{(v_i, v_{i+1}) \mid 1 \leq i < n)\}$. The path P exists in a graph G if P is a subgraph of G, written $P \subseteq G$.

Agents i and j are k-neighbors in $\mathbf{X}$ if there exists a path $P \subseteq UDG(\mathbf{X})$ from i to j where $|P| \leq k$.

978-0-7695-3553-1/08 $25.00 © 2008 IEEE
DOI 10.1109/SASOW.2008.62

Definition 2. The *k-neighborhood* of i is denoted as $N_k(i)$ and is defined as the set of k-neighbors of j, that is $\{j \neq i \mid j \text{ is } k\text{-neighbor of } i\}$, the closed k-neighborhood is denoted as $N_k[i] = N_k(i) \cup \{i\}$.

An agent moves from position x_i to position y_i by following a trajectory γ_i where $\gamma_i : [0,1] \to \mathbb{R}^m$, $\gamma_i(0) = x_i$ and $\gamma_i(1) = y_i$. Similarly, a set of agents moves from one configuration to another by following a collection of trajectories $\gamma = [\gamma_1, \ldots, \gamma_n]$, called a motion pattern.

Definition 3. A *motion pattern* between configuration $\mathbf{X}$ and $\mathbf{Y}$ is $\gamma : [0,1] \to \mathbb{R}^{m \times n}$, where $\gamma(0) = \mathbf{X}$ and $\gamma(1) = \mathbf{Y}$.

Two motion patterns γ and δ can be composed as long as $\gamma(1) = \delta(0)$, written $\gamma \circ \delta$. Furthermore we can consider fragments of a motion pattern where δ is a fragment of γ if there exists $c \in [0,1]$ and $b \in [0,1]$ such that $\delta(t) = \gamma(ct + b)$ for $t \in [0,1]$, written $\delta \subseteq \gamma$.

Definition 4. A motion pattern γ is *connected* if UDG($\gamma(t)$) is connected for $t \in [0,1]$.

In general motion patterns are used to get closer to some final goal configuration, the following definition formalizes this concept.

Definition 5. A motion pattern γ makes *progress* with respect to $\mathbf{Z}$ if $\forall i \ \ \|\gamma_i(1) - z_i\| \leq \|\gamma_i(0) - z_i\|$. If $\exists j$ s.t. $\|\gamma_j(1) - z_j\| < \|\gamma_j(0) - z_j\|$ it makes *strict progress*.

If $\mathbf{X}$ and $\mathbf{Y}$ are two connected configurations it is easy to verify there always exists a connected motion pattern γ from $\mathbf{X}$ to $\mathbf{Y}$.

Assuming the communication graph $UDG(\mathbf{X}(0))$ is connected, we are interested in producing connected motion patterns which make progress with respect to the desired configuration of the agents.

III. Connectivity Service

We suppose a synchronous setting where the agents operate in rounds, at the beginning of round k the system is in configuration $\mathbf{X}_k$ and the agents have a target configuration $\mathbf{Y}_k$ for the next round. The communication between agents is done using a reliable broadcast primitive that guarantees timely delivery of the messages to all the agents in the neighborhood. The connectivity service should produce a motion pattern γ that starts at $\mathbf{X}_k$ and makes progress with respect to $\mathbf{Y}_k$.

Internally the connectivity service may need to exchange messages with neighboring agents and perform some local computation. Initially we assume the length of the time interval between rounds is enough to allow the connectivity service to produce the motion pattern and the agents to follow the trajectories. In general we characterize the connectivity service by the properties of the motion patterns it produces.

Definition 6. The connectivity service is *safe* if it produces connected motion patterns.

While an agent is moving through a trajectory returned by the connectivity service it may need to slow down or stop unexpectedly, for example when encountering an obstacle, taking a sample or deploying equipment. Therefore a desirable property is to tolerate some number of these unpredictable events.

Definition 7. Let $\gamma = [\gamma_1, \ldots, \gamma_n]$ be a motion pattern, let Q be a subset of V and $H = [h_1, \ldots, h_n]$ be a set of functions where $h_i : [0,1] \to [0,1]$. We define the motion pattern $\delta(\gamma, Q, H)$ where $\forall i \notin Q : \delta_i(t) = \gamma_i(t)$ and $\forall i \in Q : \delta_i(t) = \gamma_i(h_i(t))$.

The motion pattern γ *tolerates f changes in velocity* if for any set $Q \subseteq V$ such that $|Q| \leq f$ and any set of H of functions the motion pattern $\delta(\gamma, Q, H)$ is connected.

A connectivity service is *robust to changes in velocity* if the motion patterns produced by the service tolerate n changes in velocity.

We cannot require a connectivity service to make strict progress without making assumptions about the target configuration, since in some cases no connected strict progress is possible.

Definition 8. Let the system be at round k and suppose that for all rounds $k' \geq k$ the agents request the target configuration $\mathbf{Y}$, where $\mathbf{Y}$ is connected. A connectivity service makes *progress* if after a finite number of rounds r it holds that $\mathbf{X}_{k+r} = \mathbf{Y}$.

IV. Intersecting Disks Connectivity Service

The service has three phases, a collection phase, a proposal phase and an adjustment phase. The first two phases involve exchanging messages with its neighbors while the third phase requires no communication.

In the collection phase the connectivity service queries the location service to obtain its current position x_i. Each agent broadcasts its position to its neighbors and simultaneously learns the positions of its neighborhood N_i.

The proposal phase assumes access to a filtering function FILTER which receives the current neighborhood N_i and returns a subset $N_i' \subseteq N_i$. Each agent calculates the region R_i defined by the intersection of a set of disks of radius r centered at the position of each agent in N_i'. The point p_i inside R_i which is closest to the target y_i is broadcast as a proposal, and the proposed neighborhood pN_i is received.

The adjustment phase uses a safety function SAFE which receives the current filtered neighborhood N_i' and the proposed neighborhood pN_i and returns a boolean. If SAFE returns `true` the service returns a linear trajectory from x_i to the proposed target p_i, otherwise it returns a linear trajectory that goes half way between the current position and the proposed target.

The correctness proof of the algorithm is subtle, but the intuition is straightforward. First, notice region R_i is non-empty, and in particular since it is the intersection of a set of disks, all of which contain x_i, it holds that $x_i \in R_i$.

Furthermore, R_i is the result of the intersection of convex shapes and it is therefore also convex. Since $x_i \in R_i$ and $p_i \in R_i$ from the definition of convexity any point between x_i

1: **Collection phase:**
2: $y_i \leftarrow$ agent's i target
3: $x_i \leftarrow query\_positioning\_device()$
4: **broadcast** x_i to all neighbors
5: $N_i \leftarrow \{x_j \mid \text{for each } x_j \text{ received}\} \cup \{x_i\}$

6: **Proposal Phase:**
7: $N_i' \leftarrow \text{FILTER}(N_i)$
8: $R_i \leftarrow \bigcap_{x_j \in N_i' - x_i} D_r(x_j)$
9: $p_i \leftarrow$ point inside R_i closest to y_i
10: **broadcast** p_i to all neighbors
11: $pN_i \leftarrow \{p_j \mid \text{for each } p_j \text{ received}\} \cup \{p_i\}$

12: **Adjustment Phase:**
13: **if** not $\text{SAFE}(N_i', pN_i)$ **then**
14: $p_i' \leftarrow x_i + \frac{1}{2}(p_i - x_i)$
15: **return** trajectory from x_i to p_i'
16: **else return** trajectory from x_i to p_i

Fig. 1. Intersecting disks connectivity service run by agent i

and p_i is also in R_i, so in particular the adjusted target must be in R_i.

Lemma 1 (Adjustment)**.** *If two filtered neighbors i and j adjust their proposals, the resulting proposals are connected.*

Proof: The adjusted proposals of agents i and j are $p_i' = x_i + \frac{1}{2}(p_i - x_i)$ and $p_j' = x_j + \frac{1}{2}(p_j - x_j)$, and the distance between them is:

$$\begin{aligned}\left\|p_i' - p_j'\right\| &= \left\|x_i - x - j + \frac{1}{2}(p_i - p_j + x_j - x_i)\right\| \\ &= \left\|\frac{1}{2}(x_i - x_j + p_i - p_j)\right\| \\ &\leq \frac{1}{2}\left\|x_i - p_j\right\| + \frac{1}{2}\left\|x_j - p_i\right\|\end{aligned}$$

Finally, since i and j are filtered neighbors $\|x_i - p_j\| \leq r$ and $\|x_j - p_i\| \leq r$, hence $\left\|p_i' - p_j'\right\| \leq r$. ∎

However, even if two agents are connected and propose connected targets, they could disconnect while following their trajectory to the target. Below we prove that under some set of assumptions, any linear trajectory between a pair of agents is robust.

Lemma 2 (Robustness)**.** *The linear trajectories of any two filtered neighbors i and j whose endpoints are connected tolerate velocity changes of i and j (proof omitted due to space limitations).*

These two lemmas are enough to prove the following corollary.

Corollary 1. *The trivial connectivity service where* FILTER *is the identity function and* SAFE *always returns* `false` *is safe and robust to velocity changes.*

Proof: Let (i, j) be an edge in the graph $UDG(\mathbf{X})$. Since FILTER returns the same set it received agents i and j are filtered neighbors, and since SAFE returns `false` both agents adjust their proposals. Therefore by the adjustment lemma the resulting proposals p_i' and p_j' are connected.

Finally by robustness lemma the linear trajectories for agents i and j towards p_i' and p_j' tolerate velocity changes of i and j. Since this holds for all edges resulting motion pattern is safe and robust to velocity changes. ∎

However this choice of functions is the worst in terms of progress and requires preserving all edges. In the next sections we study some natural choices for the FILTER and SAFE functions that lead to different algorithms with better properties, in all cases we prove the resulting service is indeed safe and robust to velocity changes.

A. No neighbor filtering

Without filtering the motion of the agents is very constrained since R_i depends on all the neighbors. However, even without filtering the connectivity service can allow edges to be removed if the predicted neighborhood is safe. Here we define a SAFE function that guarantees connectivity while allowing some edges to be lost.

Definition 9. A path $P_j \subseteq UDG(pN_i \cup \{p_i\})$ from p_i to p_j is safe if for every edge $(p_l, p_m) \in E(P_j)$ it holds that $(x_l, x_m) \in UDG(N_i \cup \{x - i\})$.

In other words a path from i to j in the proposed neighborhood of i is safe if the same path existed in the original neighborhood.

Definition 10. Let SAFE return `true` if $\forall x_j \in N_i$ there exists a safe path $P_j \subseteq UDG(pN_i \cup \{p_i\})$ from p_i to p_j, and `false` otherwise.

This is clearly stronger than connectivity of $UDG(pN_i \cup \{p_i\})$ since not only there needs to exist a path from every pair of vertices in the proposed neighborhood but the paths have to be safe.

Lemma 3. *The intersecting disks connectivity service with no neighbor filtering and the described* SAFE *function is safe and robust.*

Proof: Let i and j be any two neighboring agents. We divide the analysis in cases depending on the existence of a safe path from i to j.

IF THERE IS NO SAFE PATH. Then both agents adjust their proposal and by the adjustment lemma their adjusted proposals are connected. Since they are "filtered" neighbors and execute a linear trajectory towards connected proposals by the robustness lemma their trajectories tolerate velocity changes of either agent.

THERE IS A SAFE PATH. Let (p_l, p_m) be an edge in the safe path. Suppose that neither agent adjusts their proposals, by the robustness lemma their trajectories tolerate velocity changes of either agent and remain connected. However if l or

m decide to adjust, their resulting trajectories are fragments of the previous ones, and therefore also tolerate velocity changes.

Therefore every pair of neighboring agents in the current round are connected by a path in the next round. Moreover, the trajectories followed by each agent in the path tolerate the velocity changes of any agent in the path, hence the motion pattern is safe and robust. ■

B. Local filtering

Ideally each agent has access to some oracle that provides a subset of neighbors to maximize R_i and allow a proposed target closer to the real target, while guaranteeing that if the agent stays connected to this subset of neighbors, the graph remains connected.

To implement such an oracle would require using global information about the graph, but this demands more communication and does not scale well with network size. Here we attempt to approximate such an oracle using local information.

One approach to reduce the number of neighbors and hopefully have more slack when proposing a target is to use a connected spanning subgraph of $G(\mathbf{X})$. Examples of such graphs are the Gabriel graph (GG) [7], the relative neighbor graph (RNG) [8], and the local minimum spanning tree (LMST) [9]. These structures are all connected and can be computed using local algorithms.

Here we suppose that each agent locally computes some graph H_i, and an edge (i, j) is included in H iff $(i, j) \in E(H_i) \wedge (i, j) \in E(H_j)$, where H is some spanning subgraph of $G(\mathbf{X})$.

Definition 11. With local filtering FILTER returns the set $\{x_j \mid (j, i) \in E(H_i)\}$.

If in the proposed neighborhood the filtered neighbors are still connected, the proposed neighborhood is safe, otherwise it is unsafe.

Definition 12. SAFE returns `true` if $\forall j \in N_i'$ it holds that $\|p_j - p_i\| \leq r$.

To prove this FILTER and SAFE functions lead to a safe and robust connectivity service we use a similar argument to the one presented in lemma 3.

Lemma 4. *The connectivity service with local filtering is safe and robust*

Proof: Since H is connected, it is enough to prove that every edge of H is preserved from one round to the next using trajectories robust to velocity changes. Let (i, j) be any edge of H, therefore $x_j \in N_i'$ and $x_i \in N_j'$ so the agents are filitered neighbors.

$\|p_i - p_j\| > r$ Therefore `Safe` returns false and both agents adjust their proposal to p_i' and p_j', which by the adjustment lemma are connected. Since i and j are filtered neighbors and they propose trajectories to connected endpoints by the robustness lemma these trajectories are robust to changes in velocity of either agent.

$\|p_i - p_j\| \leq r$ Regardless of what i and j decide on the adjustment phase the endpoints of their trajectories will be connected. Finally by the robustness lemma these trajectories are robust to changes in velocity of either agent. ■

V. PROGRESS GUARANTEES.

When dealing with more than 2 agents the filtering method needs to be considered to prove progress. The problem is further complicated since not only edges are lost (which potentially could lead to more progress), but in a lot of cases edges are added.

Moreover, some motions might require loosing edges which cannot be removed by local methods, this can be handled by global cycle breaking algorithms that could be started by the service (and run in parallel) whenever no progress is being made. We will describe how to incorporate global cycle breaking routines with the algorithm in a future paper.

Even ignoring filtering issues, the greedy nature of the service means it will always try to make positive progress towards a target. However some combinations of initial and target configurations require the agents to make negative progress (move further away from its target) on the short term so that on the long term they can reach its target. Its clear the service does would never produce such motions.

VI. SIMULATION RESULTS

To evaluate the performance of the intersecting disks connectivity service we implemented a simulation framework with the different filtering routines. As the communication graph becomes dense the differences between the filtering routines become more apparent. We tested the three filtering methods described in a random connected graph where each agent has a random target $6r$ away from its initial position.

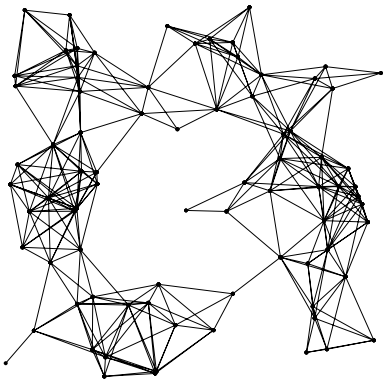

Fig. 2. Random graph with 80 agents

Figure 3 shows an arrow from the initial position of each agent to its target, dotted lines are draw between targets that are within communication range. Clearly the resulting graph is disconnected and therefore unreachable by any safe connectivity service.

We define $C(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \|x_i - y_i\|^2$ as the distance between two configurations. In this example $\mathbf{Y}$ is fixed and

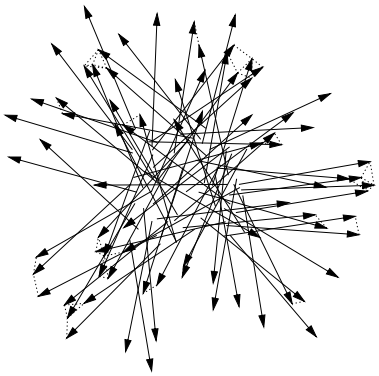

Fig. 3. The desired motion vector of each agent.

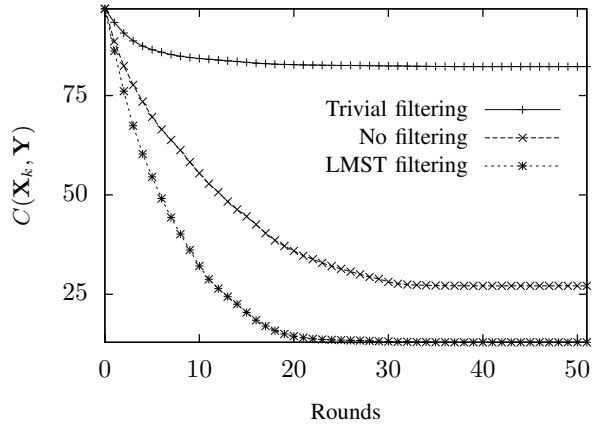

Fig. 4. Agents in a random initial position trying to reach a fixed random disconnected target using the connectivity service with three different filtering methods.

$\mathbf{X}_{k+1}$ is determined by the connectivity service while trying to advance towards $\mathbf{Y}$ from $\mathbf{X}_k$. On figure 4 we present a plot of $C(\mathbf{X}_k, \mathbf{Y})$ against k for the three different filtering methods.

A. Considering physical limitations of the agents.

So far we assumed the length of the rounds is large enough to allow the agents to move the full trajectory returned by the connectivity service. Suppose the length of the round is fixed to 1 second and assume the time required by the connectivity service to return the trajectory is negligible. Given the agent's acceleration and velocity limitations we can compute the maximum distance each agent can travel during one round, suppose all agents can travel d_{max} in one round.

Each agent has a long term target y_i, and a short term target y'_i, where where $y'_i \parallel y_i$ and $\|x_i - y'_i\| = \min(d_{max}, \|x_i - y_i\|)$. Therefore when $d_{max} = \infty$ we get the same trajectories produced when ignoring the physical limitations of the agents.

1) Column of agents moving forward.: Consider a set of agents arranged in a vertical line where $\|x_i - x_{i+1}\| = r$ for $0 \leq i < n$. Furthermore suppose the agents want to translate forward some fixed distance. For all agents except agents 0 and $n-1$ the intersection region $R_i = x_i$, and therefore they can only propose $p_i = x_i$ which means they make no progress.

However agent 0 and $n-1$ can move closer to their target by getting closer to agents 1 and $n-2$ respectively. Therefore by the next round R_1 and R_{n-2} are no longer a point, and agents 1 and $n-2$ have some slack to make progress. This effects trickles down the line from the two extremes and after $n/2$ rounds all agents have made progress.

Figure 5 shows the trajectories traced by each agent as they move forward using the connectivity service with three different values of d_{max}, a dot is drawn when the agent called the connectivity service.

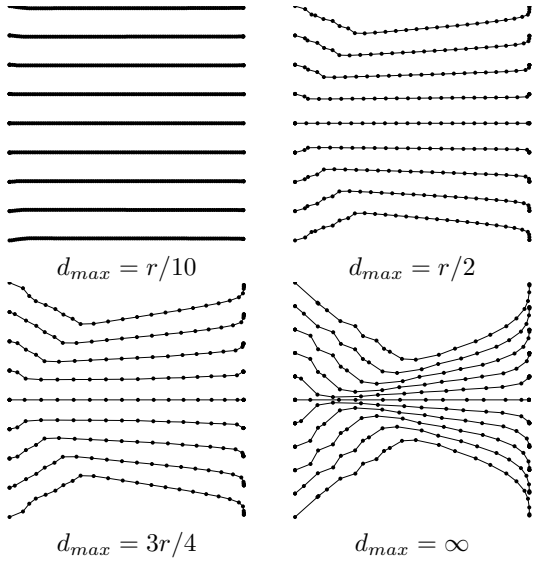

Fig. 5. Nine agents in a line configuration barely connected moving forward.

As the speed of the agents increases they cluster more while moving forward. However, the clustering could result in smaller and smaller intersecting regions R_i, since more neighbors are added. However, the LMST strategy is enough to eliminate this effect and ignores most of the newly created edges.

Figure 6 plots the distance from the agents to their target for the three different values of d_{max} used. We observe that although increasing the agents speed results in faster convergence the difference in the convergence speed of using $r/2$ and ∞ is only a few rounds.

2) Ring of agents rotating.: Consider a set of agents arranged in a ring, where agent i is connected to agent $i-1$ and agent $i+1$, for $1 \leq i < n-1$, agent 0 and $n-1$ are neighbors, and have as neighbors agent 1 and $n-2$ respectively. Moreover, assume the agents are barely connected so $\|x_i - x_j\| = r$ if agent i and j are neighbors.

Notice that unless $n = \infty$, then $\forall i$ there is some slack in R_i to allow agent i to move, but they none of them can move outward. In fact for a single agent to break the ring would require global knowledge unless n is small enough for a local

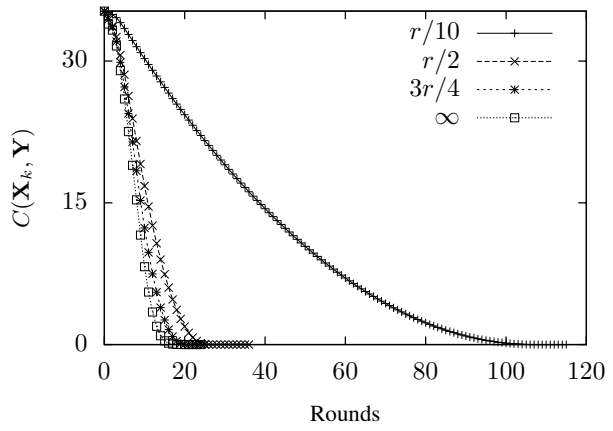

Fig. 6. Distance from target of line configuration at different speeds.

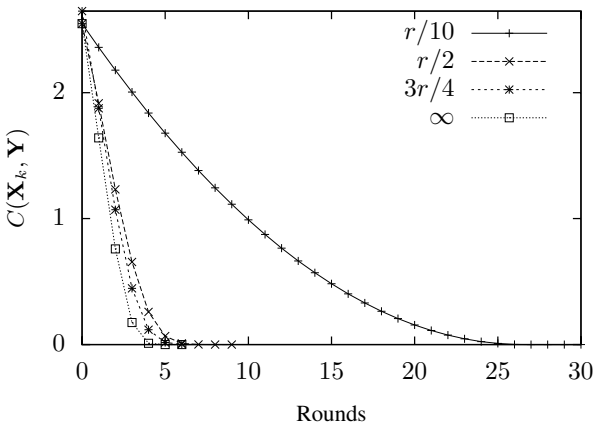

Fig. 8. Distance from target of circular configuration at different speeds.

algorithm to detect the cycle.

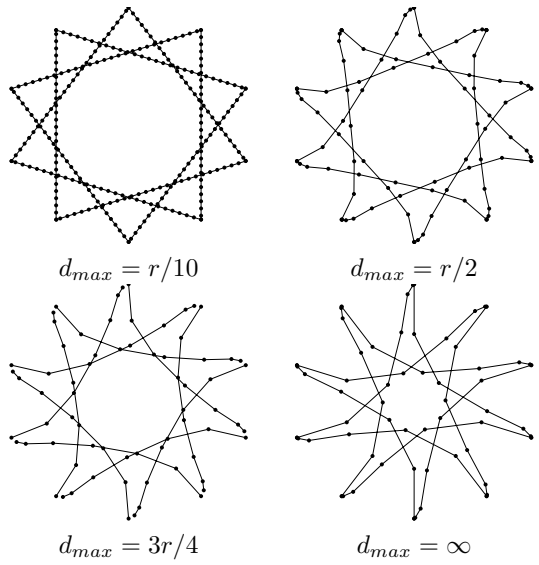

Fig. 7. Ten agents in a ring configuration barely connected rotating $3\pi/5$ counter-clockwise.

Figure 7 shows the agents around the ring rotating in place. Since the configuration is symmetrical, at each round all agents make exactly the same amount of progress. Also note that for various settings of d_{max} the agents cluster together and create new edges, here filtering was necessary to guarantee progress.

Figure 8 plots the distance from the agents to their target and presents results that resemble those shown in figure 6.

3) Grid-like and random configurations.: We invite the interested reader to visit http://people.csail.mit.edu/acornejo/research to find videos of the connectivity service being used when using more elaborate initial and target configurations. In general the service works as expected, and the relationship between d_{max}, the clustering of agents and their speed of convergence seems to hold in all cases.

VII. Conclusions and Future Work

This paper presented a distributed connectivity service that allows decoupling the task of motion planning and ensuring connectivity. However it's clear that in some cases such decoupling is not possible or may have an adverse effect on performance.

Future directions of research include designing new filtering methods for the connectivity service and developing efficient techniques for global cycle breaking. In section V we proved progress for 2 agents and we hope to close the gap by proving progress in the general setting in a future paper.

Finally the results obtained through simulation seem to point to a strong relationship between the communication speed and the speed of the agents, but this has yet to be studied formally.

References

[1] M. Schuresko and J. Cortes, "Safe graph rearrangements for distributed connectivity of robotic networks," *Decision and Control, 2007 46th IEEE Conference on*, pp. 4602–4607, 2007.

[2] M. M. Zavlanos and G. J. Pappas, "Controlling Connectivity of Dynamic Graphs," *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 6388–6393, 2005.

[3] K. Savla, G. Notarstefano, and F. Bullo, "Maintaining limited-range connectivity among second-order agents," *SIAM Journal on Control and Optimization*, 2007.

[4] M. M. Zavlanos and G. J. Pappas, "Potential Fields for Maintaining Connectivity of Mobile Networks," *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, vol. 23, no. 4, pp. 812–816, 2007.

[5] P. Bahl, J. Y. Halpern, L. L., Y.-M. Wang, and R. Wattenhofer, "Analysis of a Cone-Based Distributed Topology Control Algorithm for Wireless Multi-hop Networks," *Principles of Distributed Computing*, pp. 264–273, 2001.

[6] X.-Y. Li, Y. Wang, P.-J. Wan, W.-Z. Song, and O. Frieder, "Localized low weight graph and its applications in wireless ad hoc networks," *IEEE INFOCOM*, vol. 4, 2004.

[7] K. Gabriel and R. Sokal, "A new statistical approach to geographic variation analysis," *Systematic Zoology*, vol. 18, no. 3, pp. 259–278, 1969.

[8] G. T. Toussaint, "The relative neighbourhood graph of a finite planar set," *Pattern Recognition*, vol. 12, no. 4, pp. 261–268, 1980.

[9] N. Li, J. C. Hou, and L. Sha, "Design and analysis of an MST-based topology control algorithm," *INFOCOM*, vol. 3, pp. 1702–1712, 2003.