The Impact of Topology and Communication Models on Connectivity in Networks

Leilani Gilpin
Department of Mathematics
Undergraduate Honors Thesis 2010-2011
University of California, San Diego
Advisor: Melvin Leok

June 20, 2011

Abstract

Autonomous routing algorithms are intended to reach a global solution after nodes independently process and share information. However, the role of the mechanism used to share information has been previously overlooked in previous analyses of these algorithms and network connectivity. In this paper, I explicitly study how these network-communication models affects network connectivity. In addition, I examine properties of connectivity in robotic networks with a defined communication topology.

I show that network connectivity depends on the communication model by using a taxonomy of communication models and identifying models of interest. I will then analyze how network connectivity is preserved or not guaranteed through different models. These results are important for studying the connectivity of distributed autonomous routing protocols because certain models are best for proving model-independent conditions that guarantee connectivity, while other models are best for proving model-independent conditions that do not guarantee network connectivity.

Contents

Chapter 1

Introduction

After spending a summer at Rutgers, New Jersey, at DIMACS (Center for Discrete Mathematics and and Theoretical Computer Science) researching properties of convergence and connectivity in communication model networks, I came back to UCSD in the fall and continued looking at networks. While looking at channel reliability, and the numbers of neighboring nodes processed per time step, I began to contemplate the issue of connectivity when a network is constantly changing. In addition, I incorporated this into my research in geometric mechanics by investigating the same properties in robotic networks.

Networks, as studied in computer science, are expressed as graphs. They are a collection of nodes, which represent computers or devices, interconnected by edges that represent communication channels amongst the devices. Computer networks are used for a variety of purposes including facilitating communications, sharing hardware, and sharing software. It is also useful to classifying networks as convergent (messages are guaranteed to reach the destination d over time), or that they can oscillate. Using this notion of convergence, we can express connectivity in these networks based on their ability to converge to a destination.

Robotic networks, are represented as graphs but they include a spatial component. Geometric objects model the interaction between robotic networks with the environment and proximity graphs represent the topology of a network of robotic agent. Robotic networks can model motion coordination in vehicles, and there has recently been work using robotic networks to model motion coordination in self-organized biological groups. The main motivation for using robotic networks is to model the connectivity of different communication models. I have been focusing on the computer simulation component, by trying to generalize the connectivity in robotic networks as a topological property.

Chapter 2

Communication Model Networks

2.1 Communication Model Definition

An instance of a network is an undirected graph $G = (V, E)$ with a distinguished destination node, d and for each node $v \in V$ a set of permitted paths $\mathcal{P}_v$, which is a subset of all paths from v to d. There is also a ranking function $\lambda_v : \mathcal{P}_v \to \mathbb{N}$ indicating node v's preference for each permitted path where lower rank are more preferred paths.

The problem is to find a path assignment $\pi = \{\pi_v\}_{v \in V}$ that for each $v \neq d$, is

1. consistent - Assuming that the next step in path π_v is u, we have that $\pi_v = v\pi_u$ (if v extends a path from u to d, then that path from u to d is assigned to u.

2. connected - For all neighbors $w \neq u$ of v, $\lambda_v(v\pi_u) < \lambda_v(v\pi_w)$. Which means that π_v is more preferred than any other $v\pi_w$. We assume that $\pi_d = d$.

Let $\mathcal{C}$ be the set of communication channels that might be used. For each edge $\{u, v\}$ in the undirected instance graph, $\mathcal{C}$ contains directed channels (u, v) and (v, u). We assume that each channel is FIFO, so that messages that are written by u to the channel (u, v) and that are not dropped by the channel are process by v in the same order in which they were written.

Definition 2.1.1. Path assignments $\pi_v(t)$ *is the path to the destination that node v chooses at the end of step t. The collection of all path assignments is* $\pi(t) = \{\pi_v(t)\}_{v \in V}$. $\pi_v(t+1) = \pi_v(t)$ *unless v runs an update in step t+1 (we assume that one node is updated per time step).*

Definition 2.1.2. Known routes *After the execution of the first part of the algorithm in step t, $\rho_v(c, t)$ contains the contents of the last update that v successfully processed from channel c. $\rho_v(c, t+1) = \rho_v(c, t)$ unless v updates from channel c in step $t + 1$. We let $\rho_v(c, 0) = \epsilon$ for every $v \in V$ and $c \in \mathcal{C}$.*

Definition 2.1.3. Channel contents *For a channel c =(u,v), let c(t) be the contents of c at the beginning of step t, and let $m_c(t)$ be the number of messages in c at the beginning of step t. $c_i(t)$ denotes the ith message in c at the beginning of step t, where the first message in the oldest, so that messages are held in the channel like a queue. In addition, we let c(0) = $\emptyset$ for every $c \in \mathcal{C}$.*

In each step of the algorithm, nodes: collect information from channels, choose route objects based on their ranking function and permitted paths, and share information by writing route objects to channels. The communication models studied here only affect the first of these actions. We use an activation sequence to specify the nodes involved in each round and how the first action is executed. An activation sequence determines the algorithm's execution.

Definition 2.1.4. *An activation sequence α is a function on the on the non-negative integers that assigns to each $t \in \mathbb{Z}_{\geq 0}$ a quadruple (U, X, f, g) such that:*

1. *$U \subseteq V$ is the set of vertices that will update in step t.*
2. *$X \subseteq \mathcal{C}$ is the set of channels that will be updated in step t. For each $c = (u, v) \in \mathcal{C}$, we require that $v \in U$ so that the receiving end of each channel is one of the nodes that is updating in step t.*
3. *$f : X \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$ indicates how many messages from each channel should be processed. For $c = (u, v) \in X$, v will process $f(c)$ messages from c. If $f(c) = \infty$, then v will process all messages in the channel*
4. *$g : X \to \mathcal{P}(\mathbb{Z}_{>0})$ indicates which, if any, messages will be dropped from each channel; the elements of $g(c)$ are the indices of the messages that will be dropped. We require that if $f(c) = 0$ then $g(c) = \emptyset$, and if $0 < f(c) < \infty$, then $g(c) \subseteq \{1, 2, 3, .., f(c)\}$.*

We say that an activation sequence is fair if every node tries to read each of its channels infinitely often and, if a message is dropped from channel c (a possibility when unreliable channels are used), then there is a later message on c that is not dropped. This research focuses on fair activation sequences.

Definition 2.1.5. *Given a network instance and an activation sequence α, the iterative routing algorithm executes as follows, starting with $t = 0$.*

1. *Let $(U, X, f, g) = \alpha(t)$.*
2. *For each $v \in U$ and $u \in \mathcal{N}(v)$ such that $(u, v) \in X$*
   (a) *Let $c = (u, v)$*
   (b) *If $f(c) = \infty$, let $i = m_c(t)$; if $f(c) < \infty$ let $i = max\{f(c), m_c(t)\}$.*
   (c) *If $\{1, 2, .., i\} \backslash g(c) \neq$, let j be the largest element of this set, and let $\rho_v(c, t)$ be the route in the j^{th} message in c. If $\{1, 2, ..., i\} \backslash g(c) = \emptyset$, let $\rho_v(c, t) = \rho_v(c, t - 1)$.*
   (d) *Delete the first i messages from c, and set $m_c(t + 1) = m_c(t) - i$.*
3. *For each $v \in U$, set $\pi_v(t)$ to be the most preferred path from the set $\{\rho_v((u, v), t) \cap \mathcal{P}_v \mid u \in \mathcal{N}(b)\}$ if $v \neq d$, and let $\pi_v(t) = d$ otherwise.*
4. *For each $v \in U$ and $u \in \mathcal{N}(v)$, if $\pi_v(t) \neq \pi_v(t - 1)$ and if prescribed by export policy, write the path $\pi_v(t)$ to the channel (v, u) and increase $m_{(v,u)}(t + 1)$ by one.*
5. *Increate t and repeat the process from step 1.*

Definition 2.1.6. *An activation sequence, α* **converges** *to the path assignment π if, for the sequence of path assignments $\pi(t)$, there exists some $t*$ such that for any $t' \geq t*$, $\pi(t') = \pi$. We write $lim_{t\to\infty}\pi(t) = \pi$, and say that α converges if the limit exists.*

Definition 2.1.7. *A network model is said to be* **connected** *if there exists a path assignment that converges to the destination node, d.*

If a network model is not connected, then then there exists a node that is infinitely changing path assignments. In this case, we say that the network oscillates.

2.2 Taxonomy Definition

This work focuses on a four dimensional model space that dictates the communication model.

The four dimensional model is as follows:

1. Number of nodes updating: An activation sequence must specify the set of nodes that update at each step. For this research, we require exactly one node to update at each step. However we can have that every node updates at every step, there are no restrictions on which nodes update, or exactly one node updates at each step

2. Number of neighbors processed: Each model specifies how many channels a node should process when it updates.

   (a) **E**(**Every**): Whenever a node updates, it processes messages from every one of its neighbors. $\Rightarrow X_v = \mathcal{N}(v)$ for every $v \in U$.

   (b) **M**(**Multiple**): Whenever a node updates, it processes messages from some subset of its neighbors (potentially multiple neighbors), including the possibility of processing no channels and that of processing all channels. This imposes no additional restriction on X_v.

   (c) **1**: Whenever a node updates, it processes messages from exactly one of its neighbors. $|X_v| = 1$ for every $v \in U$.

3. Number of messages processed per channel: Each model specifies how many messages a node should read from a channel when it processes that channel.

   (a) **A**(**All**): Whenever a node v updates and is assigned to process messages from a neighbor u, v processes all of the messages in this channel so the $f_v \equiv \infty$.

   (b) **S**(**Some**): There are no restrictions on the number of messages that a node processes from each of its neighbors when it updates.

   (c) **F**(**Forced**): Each node is forced to process at least one message from each of the neighbors from which it updates.

   (d) **O**(**One**): Whenever a node v updates and is assigned to process messages from a neighbor u, it processes exactly one message from this channel: $f_v \equiv 1$. If unreliable channels are used, this message could be lost.

4. Channel reliability: Channels can either be reliable, or unreliable as defined below.

    (a) **R**(**Reliable**): Every message placed in a channel (u, v) by u is always read by v. Hence the functions g_v in the fourth component of an activation sequence entry are always identically equal to .

    (b) **U**(**Unreliable**): Some messages placed in channels are not read. Hence the functions g_v are not necessarily identically equal to .

The symbols for each option in the last three dimensions are used to abbreviate the combinations.

Definition 2.2.1. *We say that model B preserves the oscillations of another model a if the existence of a nonconvergent activation sequence α in A for some network instance I implies that there exists an activation α' in B for network instance I that does not converge. This implies that the network is not connected.*

Definition 2.2.2. *We say that a model B realizes the executions of model A (exactly, exactly with repetition, or as subsequence) if, for every network instance using B and activation sequence α (in B), there exists an activation sequence α' in the A such that, if $\{\pi(t)\}_t$ and $\{\pi'(t)\}_t$ are the path-assignment sequences induces by α and α', we have*

1. *Exact realization: $\forall t, \pi'(t) = \pi(t)$ so that the sequences are the same. We then write $A \leq B$.*

2. *Exact realization with repetition: $\exists f : \mathbb{N} \Rightarrow \mathbb{N}$ such that $\forall i, ji < j \Rightarrow F(i) < f(j)$ and $f(t) \leq k < f(t+1) \Rightarrow \pi'(k) = \pi(t)$ i.e. $\{\pi'(t)\}_t$ is obtained from $\pi(t)\}_t$ by replacing each $\pi(t)$ with one or more occurences of $\pi(t)$. We then write $A \lesssim B$.*

3. *Realization as a subsequence: $\exists f : \mathbb{N} \Rightarrow \mathbb{N}$ such that $\forall i, j,\ i < j \Rightarrow f(i) < f(j)$ and $\forall t, \pi'(f(t)) = \pi(t)$, i.e. $\{\pi(t)\}_t$ is a subsequence of $\{\pi'(t)\}_t$. We then write $A \preceq B$*

Exact realization implies exact realization with repetition, which implies realization as a subsequence. If one model realizes the executions of another, in any of the senses, it immediately follows that the first model preserves the oscillations of the second model.

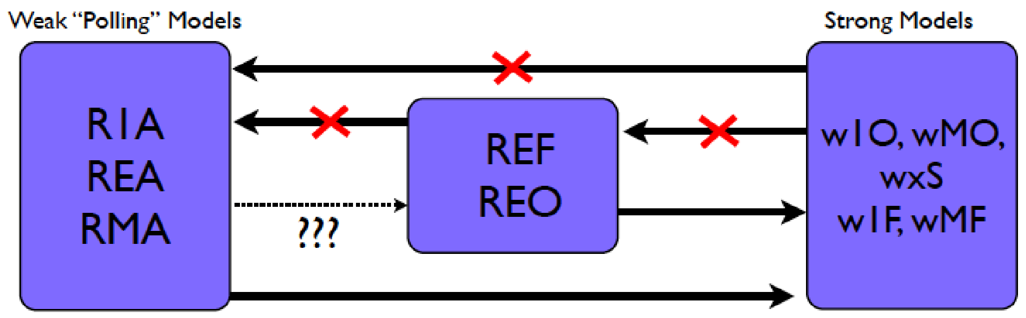

Figure 2.1: A map of the previously known relationships between network models. Arrows denote preservation of oscillation. Crossed-out arrows indicate models on the right that can oscillate in ways the models in the middle and left cannot.

2.3 Results

Previous research in this area characterized the defined taxonomy into three categories: Weak "polling" models, mid-range models, and strong models. When looking solely at reliable channels, it remained left to show the relationship between RMA (Reliable channels, updating from a multiple number of neighbors where each node processes all messages in the channel) and REO (Reliable channels, updating from every neighbor, where each node processes one message in the channel). Though it was assumed that REA (Reliable channels, updating from every neighbor, where each node processes all messages in the channel) and RMA were nearly equivalent models, I found a network that can oscillate in RMA, but it cannot oscillate in REA. Further, by showing that this network cannot oscillate in REO, there is further classification needed to categorize these models.

Using the network NAUGHTY GADGET [9], as shown in Figure 2.2, I show that there exists a network that can oscillate in some models, but it is guaranteed to converge in other models.

Proposition 2.3.1. *Naughty Gadget cannot oscillate in REO.*

Proof. Look at node 4 when it is first activated: There are three possibilities:

1. Node 4 chooses no path: Since we are working with fair activation sequences, node 4 will be updated infinitely often and it will eventually choose a path, and hence it will fall into case 2 or case 3.

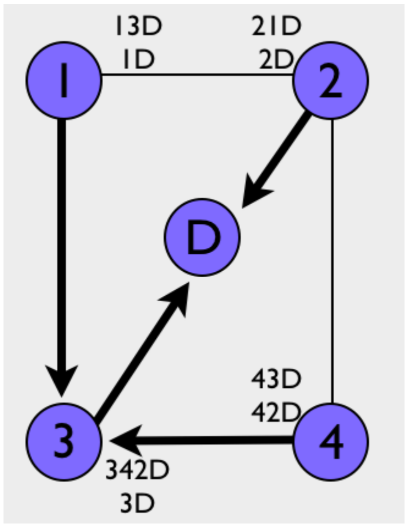

Figure 2.2: The example network NAUGHTY GADGET. This network does have a stable, connected solution, but it may oscillate when we examine different network models. This model was used to show new relationships between network models. The allowable paths are written in order from most preferred to least preferred.

2. Node 4 chooses 42D for the first time in the activation sequence: That implies that node 2 chose path 2D sometime earlier in the activation sequence. We can also assume that node 3 is activated sometime after node 4, and node 1 is activated sometime after 2 (otherwise 4 would have chosen path 43D, which would contradict our hypothesis, and 2 would have chosen path 21D, also to contradict our hypothesis). Because we have assumed fair activation sequences, node 1 will eventually have to be activated and choose path 1D. After that point, node 2 will be activated and choose path 21D. Similarly, node 3 will eventually have to be activated and it will choose path 342D, since it is its most favorable path. After this point when node 4 is updated, it will choose no route, since it is conflicted by its neighbors' path assignments. When node 3 is next activated, it will see that node 4 chose no route and therefore, it will choose path 3D. From here, the next time node 4 is activated, it will see that node 3 chose path 3D, and case 3 now applies.

3. Node 4 chooses 43D for the first time in the activation sequence: That implies that node 3 chose path 3D sometime earlier in the activation sequence. Since node 3 chose path 3D, when node 1 is next activated,

it will choose path 13D, since that is its most preferred path. And when node 2 is next activated, it will choose path 2D. Therefore, it is in a stable solution and it cannot oscillate.

□

Proposition 2.3.2. *If REA can oscillate, then REO can oscillate*

Note: This proves that REO realizes REA as a subsequence.

Proof. If there exists a fair activation sequence that can oscillate in REA, then create a fair activation sequence for REO as follows: For every node in the activation sequence, activate that node continuously until it receives the most current update from each of its neighbors This will ensure that each node in the activation sequence will see the most current update from its neighbors, and it can oscillate as REA does. □

Proposition 2.3.3. *NAUGHTY GADGET oscillates in UEA.*

Proof. We achieve the oscillation by dropping updates from node 3 to node 4 that contain the preferred path 43D at node 4, allowing only the withdrawl of 3D (or announcement of 342D) to successfully traverse the channel from node 3 to 4. This amounts to dropping every other message on that channel. Doing so satisfies the condition that a message is eventually delivered after any dropped message. No messages on any other channels need to be dropped.

In particular, consider the following activation sequence:

$$\begin{array}{rcccccccc}
t = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
.\quad U(t) = & d & 2 & 4 & 3 & 4 & 1 & 2 & 4 \\
\pi_{U(t)}(t) = & d & 2d & 42d & 342d & 42d & 1d & 21d & \epsilon
\end{array}$$

$$\begin{array}{rccccccc}
t = & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
.\quad U(t) = & 3 & 1 & 2 & 4 & 3 & 4 & ... \\
\pi_{U(t)}(t) = & 3d & 13d & 2d & 42d & 342d & 42d & ...
\end{array}$$

where the message containing from node 3 that advertises the path 3D to node 4 (permitting the path 43D at node 4) is dropped before node 4 updates in step t=12. Continued updates in this fashion will repeat the oscillation (steps 11-14 repeat steps 2-5). $\square$

Proposition 2.3.4. *$UEA \leq REO$*

Proof. As shown in Proposition 2.3.3, *NAUGHTY GADGET* has an oscillatory sequence in UEA, but as shown in prop, it cannot oscillate in REO. Thus REO does not preserve the oscillations of UEA. $\square$

Proposition 2.3.5. *UEO realizes UEA as a subsequence*

Proof. Using the same argument as Proposition 2.3.2 and applying Proposition 2.3.4, we get that $UEO \not\leq REO$. $\square$

Proposition 2.3.6. *If node 4 chooses path 43D, then NAUGHTY GADGET drops into a stable solution*

Proof. If node 4 chooses path 43D at time t-1, then node 3 was previously activated and chose path 3D at some t' such that $t' \leq t-1$. Now there are 4 cases at time t':

1. $U(t') = 4$: Node 4 will still choose path 43D since it is in its most favorable path. When node 3 is next activated, it will choose path 3D. Assume for sake of contradiction that node 3 choose path 342D. That would imply that node 4 previously chose path 42D, at a time when 3 did not choose path 3D. But that is a contradiction because we assumed that node 3 choose path 3D at a previous time. Therefore, our initial assumption is false and node 3 chooses path 3D as required. Because node D chooses path 3D, when node 1 is next activated, it will choose path 13D since it is in its most favorable path. Finally, whenever node 2

is activated, it chooses path 2D because it is the only valid assignment. I claim that any sequence of nodes activated after time t' will not change from these assignments, and hence, the network is in a stable solution. The only nodes that will want to change assignment are node 2 and node 3 since they are not in their most favorable solution. However, node 2 will change paths only if node 1 chooses path 1D. But since node 3 chose path 3D and node 1 chooses path 13D to be in its most favorable path, node 1 will not choose path 1D, and therefore node 2 will remain to choose path 2D. Node 3 can change to 342D only if node 4 chooses 42D. Though 42D is a valid assignment for node 4 to choose, it will not choose path 42D because path 43D is also a valid assignment and its most favorable assignment. Therefore, any sequence of nodes activated after time t' will not change to different path assignments, so the network is in a stable solution.

2. $U(t') = 3$: Node 3 will not change its path since node 4 chooses path 43D so there is no option for node 3 to choose 342D. As case 1 states above, whenever node 2 and node 1 are activated next, they will choose paths 2D and 13D respectively, and therefore the network is in a stable solution.

3. $U(t') = 1$: If node 1 is activated next, then it will choose path 13D, and when node 2 is activated after that, it will have to choose path 2D. Neither node 4 nor node 3 will change its path assignments since both nodes have chosen their most favorable paths. Therefore, the network is in a stable solution.

4. $U(t') = 2$: There are two scenarios:

    (a) Node 2 chooses path 21D: This implies that node 1 was previously activated and chose path 1D. If 1 chose path 1D, this implies that node 3 did not choose path 2D. However, this is a contradiction since it is assumed that node 3 choose path 3D. Therefore, node 2 cannot choose path 21D if node 4 chose path 43D.

    (b) Node 2 chooses path 2D: Then when node 1 is next activated, it had chosen path 13D. Now the network is in the stable solution.

□

Proposition 2.3.7. *NAUGHTY GADGET will not oscillate in REA.*

Proof. I will show that node 4 does not oscillate. There are 3 cases:

1. Node 4 chooses no route: Since neighboring nodes 2 and 3 update infinitely often, node 4 will process updates from nodes 2 and 3 and choose a route.

   (a) If node 3 is activated before node 2, then it will choose path 3D since node 4 has chosen no route. When node 4 is next updated, it will choose path 43D and it will drop into a stable solution as shown in Proposition 2.3.6.

   (b) If node 2 is activated before node 3, then it will either choose path 21D or path 2D. If it chooses path 2D, then the next time that node 4 is updated, node 4 could choose 42D, which will not oscillate by case 3. If node 2 chooses path 21D, then node 4 will eventually get a route after node 3 gets activated an chooses path 3D. Therefore node 4 will choose a path when it is activated after all its subsequent neighbors are activated and choose routes.

2. Node 4 chooses path 43D infinitely often: This will drop into a stable solution by Proposition 2.3.6.

3. Node 4 chooses path 42D infinitely often: This will go into a stable state by Proposition 2.3.8 below.

Therefore, REA cannot oscillate in REA. □

Proposition 2.3.8. *If node 4 chooses path 42D in NAUGHTY GADGET in REA, the network will go into a stable state.*

Proof. Assume that node 4 was activated at time t, so that node 2 was activated and chose path 2D at some time $t' \leq t$. I claim any arbitrary activate sequence at time $t+1$ will go into a stable state

When node 3 is next activated, it will choose path 342D, since that is its most favorable path. When node 1 is next activated, it will choose path 1D, since it is its only valid path. However, the next time that node 2 is activated, it will choose path 21D, since that is its most favorable path. Consequently, this path assignment will put the network into a stable state where it cannot oscillate. When node 4 is next updated, it will choose no route because it cannot send messages to either of its neighbors by their chosen paths. Therefore, when node 3 is next updated, it will choose path 3D, since it processes that node 4 chose no route. And finally, when node 4 is updated next, it will choose path 43D, and by case 2 in Proposition 2.3.7, it will be in a stable solution. □

Proposition 2.3.9. *NAUGHTY GADGET has an oscillatory activation sequence in RMA.*

Proof. RMA is a model like REA, except that when a node updates, it processes messages from some subset of its neighbors. (In REA, a node processes messages from every one of its neighbors). Constrain node 4 to only process messages from path 342D. (So node 4 will only receive messages from node 2, and it will never choose path 43D). If the channels that are activated at each step are given by $X(t)$, then we can have a cycle as follows:

$t =$	1	2	3	4	5	6	7	8	9	10	11	12	...
$X(t) =$	$\{(d,d)\}$	$\{(d,1)\}$	$\{(d,1),(1,2)\}$	$\{(d,1),(1,2),(d,3)\}$	{(3,1),(2,1),(d,3) }	$\{(3,1),(d,2),(d,3)\}$	$\{(3,1),(d,2),(d,3),(4,2)\}$	$\{(3,1),(d,2),(4,3),(2,4)\}$	$\{(d,1),(d,2),(4,3),(4,2)\}$	$\{(d,1),(1,2),(4,3),(2,4)\}$	$\{(d,1),(1,2),(4,3)\}$	$\{(d,1),(1,2),(d,3)\}$	...
$\pi_1(t) =$	$--$	$1d$	$1d$	$1d$	$13d$	$13d$	$13d$	$13d$	$1d$	$1d$	$1d$	$1d$	...
$\pi_2(t) =$	$--$	$--$	$21d$	$21d$	$21d$	$2d$	$2d$	$2d$	$2d$	$21d$	$21d$	$21d$	...
$\pi_3(t) =$	$--$	$--$	$--$	$3d$	$3d$	$3d$	$3d$	$342d$	$342d$	$342d$	$342d$	$3d$	...
$\pi_4(t) =$	$--$	$--$	$--$	$--$	$--$	$--$	$42d$	$42d$	$42d$	$42d$	$--$	$--$	...

Because 4 can only choose path $42D$, (or no path at all), the network oscillates between 2 valid assignments. One is where 4 chooses no path, and node 2 is in its most favorable path, and the other is where 4 chooses $42D$ and 3 is in its most favorable path. If 4 could process an update from 3, then we would drop into a stable assignment, and the network would not oscillate. In any step in the activation sequence illustrated above, at least one node, call it x, wants to switch to a more favorable path, therefore, when x is next activated, it choose a more favorable path, and the network path assignments

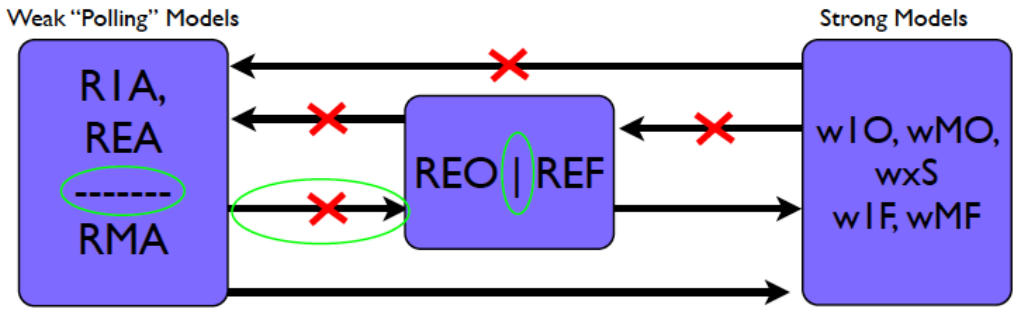

Figure 2.3: A map indicating updates to the previously known relationships between network models with reliable channels. Arrows denote preservation of oscillation. Crossed-out arrows indicate models on the right that can oscillate in ways the models in the middle and left cannot. Green circles designate my contributions

will change. Since at least one node will want another path assignment, it will oscillate, then the arguments of NAUGHTY GADGET apply and we can get an oscillation. □

Though it was previously assumed that models REA and RMA were in the same model category, the example network NAUGHTY GADGET shows that RMA can oscillate in ways that REA cannot. With this statement, REA and RMA are no longer in the same category, and further REO and REF are not in the same category. Using this knowledge, we are now working on creating new categories for these models since the relationships are more complicated than what was previously assumed.

Chapter 3

Robotic Networks

3.1 Definition

Robotic networks are a group of robots that can sense their own position, exchange messages according to their communication topology, process information, and control their motion [1]. The following the basic definition of a robot and a model for movement:

A **model robot** is a continuous-time continuous space dynamical system that is a tuple (X, U, X_0, f), where X is a d-dimensional space chosen from $\mathbf{R}^d$, $\mathbf{S}^d$ and the Cartesian products $\mathbf{R}^{d_1}$ x $\mathbf{S}^{d_2}$, for some $d_1 + d_2 = d$ called the state space. U is a compact subset of $\mathbf{R}^m$ containing $\mathbf{0}_m$ called the input space. X_0 is a subset of X, called the set of allowable initial states. And $f : XxU \to \mathbf{R}^d$ is a continuously differentiable control vector field on X, that is , f determines the robot motion x : $\mathbf{R}_{\geq 0} \to X$ via the control system

$$\text{x(t)=f(x(t),u(t))},$$

subject to the control $u : \mathbf{R}_{\geq 0} \to U$.

$x \in X$ and $u \in U$ are referred to as the physical state and an input of the robot. Usually, the physical state will have some knowledge of location, or location and velocity.

Definition 1. *A* **robotic network** *S, consists of a tuple (I, R, E_{cmm}) where $I = \{1, .., n\}$, I is called the set of unique identifiers. $R = \{R^{[i]}\}_{i \in I} = \{(X^{[i]}, U^{[i]}, X_0^{[i]}, f^{[i]})\}_{i \in I}$ is a set of mobile robots. And E_{cmm}, the communication edge map, is a map from $\Pi_{i \in I} X^{[i]}$ to the subset of I x I.*

In order to show connectivity in a robotic network, we create a set cover on the robotic network. If the distance between any two robots is covered by a circular segment, or a set of circular segments illustrating their communication range, then the robots are said to be connected. If the network is entirely covered (ie every robot is connected to every other robot), then the network as a whole is connected.

Unlike communication models in networks, robotic networks assume connectivity initially. If we were to strip this assumption, how could we generalize the notion of a connected network from an arbitrary initial state? In the next section, I discuss a probable tool for maintaining and testing network connectivity.

3.2 Connectivity in Robotic Networks

Because connectivity is a topological issue, I have been looking at ways to compute the topology of a given robotic network. The Mayer-Vietoris Sequence [5] can compute algebraic invariants of topological spaces. If we look at a robotic network as a sequence of subspaces (such that each subspace contains a strongly connected group of robots), we can apply the Mayer-Vietoris Sequence to relate the homology groups of the space to the homology groups of the subspace.

Using the definition of the mathematical definition of a robotic definition in [6] , we can partition the set of robots, S $\subset$ x-y plane. To apply the Mayer-Vietoris Sequence, choose a pair of subspaces, A, B $\subset$ X, where X is the union of the interiors of A and B, then the Mayer-Vietoris Sequence has the form:

$$\ldots\to H_n(\mathrm{A} \bigcap \mathrm{B}) \to_{\phi} H_n(\mathrm{A}) \oplus H_n(\mathrm{B}) \to_{\psi} H_n(\mathrm{X}) \to_{\delta} H_{n-1}(\mathrm{A} \bigcap \mathrm{B}) \to \ldots \; \ldots \to H_0(\mathrm{X}) \to 0$$

After studying several "basic" applications of the sequence to spheres and Mobius strips, I started contemplating the sequence's pertinence to robotic networks. Even if we could apply the Mayer-Vietoris Sequence to a set generalization of a network, could this structure truly model the connectivity of a continuously changing network?

Topology objects are appropriate to study when examining connectivity in a robotic network. However, it is difficult to apply the Mayer-Vietoris Sequence to a robotic network because the network dynamics and communications between the robots are constantly changing. Since the Mayer-Vietoris Sequence relates the homology group of any space to the homology group of two of its subspace and their intersection, these spaces need to stay constant.

But most robotic networks change communication connectivity with their neighbors over time, and the connectivity of the network usually changes constantly. Therefore, I claim that using the Mayer-Vietoris Sequence is not entirely applicable to distributed robotic networks because the set properties are constantly changing, and therefore the group homology (the key component in using the Mayer-Vietoris Sequence), will be constantly changing, and so applying the Mayer-Vietoris Sequence is not a realistic tool to determine connectivity.

Chapter 4

Conclusions

Using the taxonomy of communication models, I have generalized convergence and connectivity in networks. I defined realization relationships and classified models based on their oscillation behavior.

Though it was previously assumed that some communication models were nearly equivalent to other models, by using the example network of NAUGHTY GADGET [9], network connectivity depends on the communication model in nontrivial ways. For some pairs of models, any execution of one model can be realized as an execution of other models. Conversely, as shown in the results, we can show by example that some executions in some models cannot be realized in other models so that there are network instances which are connected in one model but oscillate in another model.

When looking at robotic networks in particular, it is difficult to model connectivity when initial assumptions about the communication of the network are overlooked. Because communication connectivity with neighboring nodes changes rapidly over time, it is difficult to generalize the connectivity of the network as a whole. In addition, it is difficult to apply topological objects to generalize notions of connectivity because of the changing environment of the system.

Chapter 5

Future Work

I am continuing to work on this project to try to answer some of the remaining questions of this classification. We have no extensively studied models in which multiple nodes are activated simultaneously. There is also the question of algorithm behavior in the context of mixed channels.

I am specifically concentrating on the role of unreliable channels. Although unreliable channels model reliable channels, (hence some results for unreliable channels do apply to a mixture of reliable and unreliable channels), we do not have results when nodes pool and others act on messages. Finally, the question of behavior in the presence of adversarial or unfaithful nodes is important. We assume that nodes use the most recent information that they have to execute the algorithm when activated; however, in adversarial settings, nodes may use the other messages to read to act in different ways that affect network connectivity.

Bibliography

[1] F. Bullo, J. Cortes, S. Martinez. *Distributed Control of Robotic Networks: Mathematical Approach to Motion Coordination Algorithms.* Princeton Univ. Press, Princeton, 2009.

[2] F. Bullo, R. Carli, P. Frasca. *Gossip Coverage Control for Robotic Networks: Dynamical Systems on the Space of Partitions.* December 28, 2009, available at `http://arxiv.org/pdf/0903.3642v2.pdf`

[3] S. Martinez, F. Bullo, J. Cortes, E. Frazzoli. *Synchronous Robotic Networks and Complexity of Control and Communication laws.* February 1, 2008, available at `http://arxiv.org/pdf/math/0501499v1`

[4] S.I. Roumeliotis, M.J. Mataric. *"Small-World" Networks of Mobile Robots*, American Association for Artificial Intelligence, available at `http://www-users.cs.umn.edu/~stergios/papers/aaai00_small_world.ps`.

[5] A. Hatcher. *Algebraic Topology.* Cambridge University Press, Cambridge, 2002. Available at `http://www.math.cornell.edu/~hatcher/AT/AT.pdf`.

[6] S. Martinez, J. Cortes, F. Bullo .*Motion Coordination with Distributed Information.* IEEE Controls Magazine, August 2007.

[7] J. Cortes .*Distributed Motion Coordination of Robotic Networks.* CONNECT: Frontiers in Science and Technology. February 18, 2010.

[8] A.D. Jaggard, V. Ramachandran, R. Wright .*The Impact of Communication Models on Routing-Algorithm Convergence.* Proceedings of ICDCS 2009. June 2009.

[9] T. G. Griffin, F. B. Shepherd, and G. Wilfong. *The Stable Paths Problem and Interdomain Routing.* IEEE/ACM Transactions on Networking, Vol. 10, No. 2, April 2002.