Thesis

Joris Z. van den Oever

May 18, 2016

Abstract

To be written

Contents

1 Introduction

2 Problem Statement

Robotic systems have been getting more mobile and prevalent in recent years. And with that comes new possibilities and uses. One project trying to take advantage of this is the TRADR project, Long-Term Human-Robot Teaming

for Robot-Assisted Disaster Response[2]. It aims to create teams of robots and humans that work together to provide incidence response. Research into robot team communication is what prompted this research.

Autonomous robots in in disaster areas, including the industrial areas given by the TRADR use cases, need to be able to cooperate amongst themselves and with humans[2]. However because of the mobile nature of such robots this raises new questions about what happens when communication becomes unreliable. Wireless signals may not reach everywhere[3, 1] and wired communication isn't generally feasible when staying mobile.

This requires us to look closer at the effects of having communication fail. While it is easy to presume that this will have undersired effects it is hard to predict what effects that will be. Only once these effects, if they exist, have been measured we can look at potential mitigation strategies. And of course determine their effectiveness.

2.1 Research Questions

To this end we will provide answers to the following questions:

- What are the effects of communication failure of multi-agent team effectiveness?
- If there are negative effects to the effectiveness, how can we mitigate these effects?

Here we define a single instance communication failure as the recipient not having a message that the sender meant for it to have. Irregardless of where in the transmission process the failure occured, e.g. while sending or receiving. And effectiveness is both the rate of task completion, and the time taken to complete the task.

2.2 Research approach

To gather the data we will create simulations of agent teams enduring communication failure. How this is set up exactly is descriped in section 3. However this will be done in two parts. The first part will be an exploratory one where pre-existing agent teams for BW4T will be used. These agents have been made for other purposes and will help identify potential stumbling blocks. The second part is a set of specifically designed agents to more closely evaluate the effects of different kinds of communication.

The differences in effectiveness from the second part will be used to see if the effects differ depending on how the communication fails and how well different agent strategies deal with the failure. This will all be simulated as it allows for stricter control of the experimental dimensions in addition to making it easier to perform a large amount experiment runs for each scenario. Afterwards the results of the simulations can be compared to baselines where no communication failure is happening to answer the first research question. Then based on the

data we can see where the effects are strongest, and if there are agents that work well in different situations. From these we can identify potential mitigation strategies. Which can be simulated in the same scenarios as before to compare their effectiveness and answer the second question.

The gathered data will be analysed using Chi square analysis to determine if the results are significantly different from the reference data. This will give us two analyses using both the one agent team baseline and the no communication team. This comparison we will only look at the rate of task completion within the timeout period.

If this shows there are indeed significant differences we can look for the effect the different communication failure scenarios on task completion time. As a first step regression analysis can see if there is an effect using both the type of team, size of the team, and the failure scenarios as independent variables. Pending determination on if they have a signicant effect on the task completion rate. As an additional analysis to if there is a difference in how the different team types are affected each team type will have a separate regression analysis using the team size and failure scenarios. The same analysis will be done using the task completion time as the dependent variable.

3 Experiment Design

To determine the effects of communication failure, if any, on multi-agent systems we will use a well explored domain for multi-agents systems. The Blocks World for Teams GOAL environment as this is well defined and thoroughly explored system for doing research on multi-agent systems.(references) The basic concept is that an agent, or multiple agents, have to bring blocks to a certain place, the dropzone in a specific order. The blocks are distinguished by color only so any block that is the right color will suffice. These blocks are found in rooms that have one entrance from the hallway spaces. While that is the basic scenario it has built in several option for adjusting various environment variables such as the room placement

3.1 Experiment Dimensions

However it is not just the room placement that can be varied. We have categorised the variables involved in the above mentioned experimental setup. Each categorgy has multiple dimensions as shown in Table 1 and will be discussed in more detail.

The environment is mainly the way the experiment world is constructed, how many hallways and where, and the placement of the rooms, but also includes the block sequence and agent collisions.

Second is the agents and their teams themselves. The team size can be adjusted, limited by the amount of computing power available. Though world limitations, such as only one agent can enter a room at a time, might limit the effectiveness of a team larger than the amount of rooms available or than

Environment	World size and room placement	Block sequence	(No) Agent collisions
Agents	Team size	hetero-/homogenous teams	Agent type
Communication	Send/receiving failure	Failure chance	(No) global percepts

Table 1: Experiment Dimensions

there are blocks in the sequence. Once you have a team you can consider team composition. Are all agents the same or do they have different programs. Finally, what kind of programs do the agents run with. Minimal programs, full coordination, what kind of data do they share, which resolution strategies are used, there is an almost infinite amount of possibilities here.

Finally there is the communication and how it can fail. Does the sending or receiving fail, and if it does fail is it a fixed percentage of messages that fails? Of does the failure rate increase over time? is the message more likely to fail if the receipient is further away? Those are several ways the failure of messages can be done. Finally there is something to consider regarding communicating itself. Do the global percepts provided by the environment always arrive? Since they include things like when a block is delivered at the dropzone. This could effectively be a side-channel for certain kinds of messages that always succeeeds. So the agents can not use it and have to distribute such information through the normal channels.

3.2 Experiment reduction

The above mentioned experiment dimensions have too many options to realistically test everything. However we have evaluated all of them to see where the dimensions can be reduced so it only requires a manageable amount of simulations. The dimensions are discussed grouped by category as given by table 1.

Environment

The size and shape of the world the agents move around in for example can be kept consistent and seen as outside of the scope for these experiments. As that would be looking at the effects of the environment on communication failure. As such only one map design will be used across the experiments.

However switching the sequence between random and only one colour is still a possibility left to us. However all the same colour, would eliminate the need to communicate about where in the sequence the team is. Which, for some types of agents, is the only thing communicated. As such using random, where it can not be assumed the communication isn't needed, is the preferred case. While the simplicity that can be assumed with just one colour might help in building the agents the increased communication if that isn't the case is preferred.

Agents

There are several components to the agent dimension. The ones identified are the size of the agent teams, the team composition, and the agent programming itself. While Communication3.2 is about how we modify the communication for the experiment, the components of the agents will decide how the communication will go, and how much of it there is.

Team size Limiting the sizes used for the agent teams would drastically reduce the experiment space as well. And not all sizes need to be tested to get a sense of the effects, if there is one, communication failures might have. The case of having only one agent, would not need a lot of experimentation as there is nothing to communicate with, however it will still be useful to do once as a baseline reference. Next, using a size sample of three agents will be useful to test, as this is the minimum amount of agents needed to have a team where it is possible to communicate with more than one agent at the same time. Which is needed for some communication failure modes to matter. Next would be five agents, as this is a larger group, but is also just a bit more than half the amount of rooms available in the default map setup and one less than the amount of blocks needed in the sequence. This means the way tasks get divided amongst agents becomes important. As not every agent can continue exploring after just one set of rooms. As well as increasing the amount of agents involved in communicating. The next step should not go too far, as experience with the environment has shown that 10 agents start to tax the hardware and might degrade the simulation. But looking at a group with a size close to the amount of rooms available will be interesting to see how that influences the behaviour. As such going one agent fewer than the amount of rooms in the standard setup, eight, has been chosen as the last group size.

Team composition All these agent teams still need a team composition. While they could have different programs, this might introduce higher order effects that would make analysis more complicated, as well as requiring a lot more simulations. While these effects might be interesting, determining the effects, if any, in the simple case takes priority. So in the interest of limiting the time spent with simulations, and analyzing homogenous agent teams will be used.

Agent Program But that still leaves options for varying the teams themselves, though they will all have similar components as shown in figure 1. These agent programs are chosen from all possibilities because they are simple, build upon each other, and let us explore what happens with differing degrees of communication. To properly compare the teams there must be a baseline.:

- A single agent randomly visiting unexplored rooms and remembering which block can be found where. It delivers blocks to the dropzone as soon as it knows of the currently needed block. After having delivered a block it

can check if it already knows of the next needed block and deliver it right away. If it does not it continues visiting random unexplored rooms.

By having one program that always exists of one agent will let us evaluate how teams perform compared to a single agent, even with the communication failure.

This agent is then expanded upon to make the teams with different strategies:

- A team of the baseline agents that now includes the ability to communicate when the current block in the sequence has been delivered. If an agent hears the block they were planning on delivering has already been delivered they drop it in a room the next time they enter it. However because of that agents can have outdated information on the location of blocks. This is updated when they enter a room, but if all blocks of the needed colour have been moved out of their rooms the agents have nothing to do anymore and will block access to rooms that other agents might need. (#Discuss with Chris/Koen: since this will not cause a deadlock in the perfect communication scenario if does this need to be addressed. Or possible mitigation strategy since it can cause deadlocks if blocks miss a message of the current block needed being delivered.)

- The above team is extended to communicate which blocks can be found in the rooms they enter. This also updates the location of blocks that got picked up and dropped in a different room. The agents also share if they are holding a block. If a block is being held by an agent the others will not try to find and pick it up.

The last set of agent teams is two ways to include coordination to the the last agent team described. The agents will coordinate their actions to prevent more than one agent doing the same task and will work ahead. This is done by picking up not just the current block in the sequence but also picking up up the following blocks in the sequence. To be able to coordinate this the agents also communicate if they intend to pick something up. So that others wont go for that colour in the sequence. But the way the teams decide who picks what block differs:

- The reactive team first asks the other agents for confirmation on their action. If another agent wants to do the same thing, the agent with the name that compares as larger gets to do it. This is achieved by having an agent with the same plan saying no if it fulfills the name order condition.

- The proactive team starts doing a task and perhaps does work unnecessarily. And then verifies that nobody else is doing it. If the same name order condition as for the other team holds, and the agent als wants to that block in the sequence it returns a message telling it to stop.

(UPDATE THE DESCRIPTION OF THE FINAL AGENT TEAMS. It only looks at the colour of the blocks for the lookahead code. If a colour is planned on being picked up for more agents than is desired in the next X blocks, where X is the amount of agents, only then will they object.)

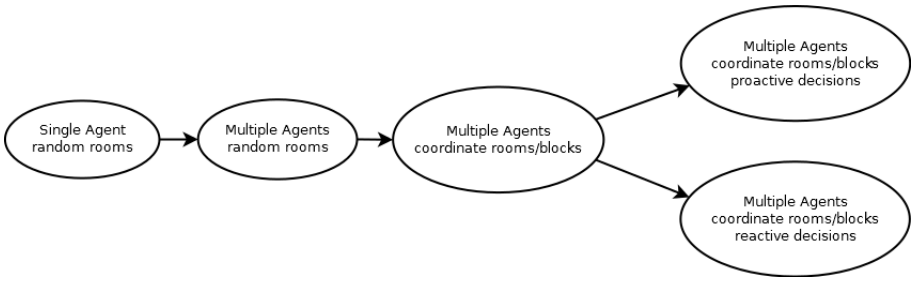

Figure 1: Agent types, each agent team has the capabilities of the previous one in the graph

Communication

For the experiment it is pretty important to figure out what kind of communication failure is going to be tested. The simplest of which is to test the difference between sending failure and receiving failure. As sending failure means that if a message fails no agent gets it but receiving failure means that some part of the team may receive it. So both options will be tested.

This can be done with various ways of determining the likelihood that a message will fail. For reference each team will also do simulations without communication failure, meaning 0%. To get insight into how different amounts of failure affect performance using set percetages is very useful. This will be a spread from 0 to a100%. To limit the amount of runs needed 25% increments will be used initially. However all communication and no communication could have different results from only a little or a lot of communication 5% and 95% will be tested as well. Making the complete range of tested percentages $0, 5, 25, 50, 75, 95$, and 100%

Finally, how are global percepts treated. These function similar to a second, infallible communication channel. As this is all about testing the effects of communication failure these will be ignored. Instead for the current state of the sequence has to be communicated.

For half of the tests the agents can still see the state of the sequence when they enter the dropzone. This allows for correcting the sequence state as the exploratory data has shown that without the correction programs break down fast otherwise. Even a 5% failure can in many cases cut the amount of successful runs in half. An overview of these results can be found in table 3.

3.3 Experiment Setup

With the above dimensions the experiment will be set up under the following conditions:

- All the simulations will be run on the same machine.

Success Guaranteed	Could fail
-	sequenceUpdate broadcast fails: wrong blocks de
room/block information broadcast fails: slows down	-
Agent Intention Broadcast: slows down	-
-	Agent plan negotiation message failure: Might wait fo

Table 2: Table of the effects communication failure might have. On the left success is guaranteed if given time. On the right situations may arise where success is no longer possible.

- The simulations will be run on a dedicated machine that only has the operating system running in addition to the simulations.
- Only one simulation is run at the same time.

These conditions are to prevent the available processing power on the system from influencing simulations between runs. And if there is only the one simulation running on the system other processes can not starve it either.

These simulations will be run through a python script to automate the entire process. It will gather and group the logs of each set of simulations with the same values for the experiment dimensions. Every set of runs is a new instance of the BW4T server to which a new instance of the runtime connects. This runtime will be run with only one mas file provided, a timeout, and the repeat amount. The timeout and repeat will be the same between sets. The resulting data will be analysed after being converted into a csv file as described in section 2.2.

4 Analysis

Table 2 gives us an overview of what was expected to happen in the various scenarios. While even the guaranteed successful scenarios might not finish within the timeframe given for the experiments they should more often do so than in scenarios where success might not always be possible. In these cases different strategies can be compared for how much they slow down compared to the reference and what the major causes for slowdown are.

The types of communication that can cause failure do this for different reasons. The first does this by incorrect information, the place in the sequence, leading to wasting resources and making the task impossible. The second one does it by having the agent logic deadlock and cause them to not do anything. It should be noted that in the second case it is theoretically possible to still finish as opposed to the first situation. {{Does this mean anything? Like make it more important to fix as it is only limited by programming?

5 Mitigation Strategies

6 Future Work

References

[1] Cisco Systems Inc. 20 Myths of Wi-Fi Interference: Dispel Myths to Gain High-Performing and Reliable Wireless , 2007. http://www.cisco.com/c/en/us/products/collateral/wireless/spectrum-expert-wi-fi/prod_white_paper0900aecd807395a9.pdf [Online; accessed 10-Februari-2016].

[2] Ivana Kruijff-Korbayová, Francis Colas, Mario Gianni, Fiora Pirri, Joachim de Greeff, Koen Hindriks, Mark Neerincx, Petter Ögren, Tomáš Svoboda, and Rainer Worst. Tradr project: Long-term human-robot teaming for robot assisted disaster response. *KI-Künstliche Intelligenz*, 29(2):193–201, 2015.

[3] D Micheli, A Delfini, F Santoni, F Volpini, and M Marchetti. Measurement of electromagnetic field attenuation by building walls in the mobile phone and satellite navigation frequency bands. *Antennas and Wireless Propagation Letters, IEEE*, 14:698–702, 2015.

A Goal Agents Source

The agent teams each have their own main module. However a lot of the other modules are shared as they share capabilities. First the different agents are detailed. Then the shared modules and knowledge.

A.1 No Communication Agent

This agent does not communicate and serves as the basis for both the single agent team and the no communication multiple agents team. The only difference between the two is the amount of agents instantiated by the .mas file.

```
1 use robot as knowledge.
2 use robot as actionspec.
3 use updateSequence as module.
4 exit=nogoals.
5 module main {
6         %drop blocks in the dropzone when they are needed
              , and communicate this.
7         if bel(in('DropZone'), holdingNextBlock) then
              putDown + updateSequence.
8
9         %drop blocks in rooms when they are not needed
              anymore
```

```
10          if bel(in(Loc),Loc\='DropZone', not(
                holdingNextBlock)) then putDown.
11
12          %go to goal places
13          if a-goal(in(Place)) then goTo(Place).
14
15          %go to goal blocks
16          if a-goal(holding(BlockID)), bel(in(Place), block
                (BlockID,_,Place), not(atBlock(BlockID))) then
                 goToBlock(BlockID).
17
18          %pickup goal blocks
19          if a-goal(holding(BlockID)), bel(atBlock(BlockID)
                ) then pickUp.
20  }
```

robotEvents.mod2g

```
1   use robot as knowledge.
2   use robotGoals as module.
3   use updateSequence as module.
4
5   module robotEvents {
6           %-------communication updates & conclusions about
                 other agents-------
7           forall bel( send('allother', Y) ) do allother.
                send(msg(Y)) + delete(send('allother', Y)).
8
9           % deduce information based on deliveries of other
                 agents:
10          forall (Agt).sent(msg('deliveryDone')) do
                updateSequence.
11
12          % Update the agent's state of movement.
13          forall bel( state(State)), percept(state(NewState
                )) do delete( state(State) ) + insert( state(
                NewState) ).
14
15          % Record when we are entering or leaving a room.
16          forall percept(in(Place)) do insert( in(Place) ).
17          forall percept(in(Place)), bel( not(visited(Place
                )), room(Place) ) do insert( visited(Place) ).
18          forall percept(not(in(Place))) do delete( in(
                Place)).
19
20          % Discover new blocks
```

```
21          forall percept(color(BlockID, ColorID)), bel( in(
                Place), not(block(BlockID, ColorID, Place)) )
                do insert( block(BlockID, ColorID, Place) ).
22
23          % Record atblock location of agent
24          forall percept(atBlock(BlockID)) do insert(
                atBlock(BlockID)).
25          forall percept(not(atBlock(BlockID))) do delete(
                atBlock(BlockID)).
26
27          % Record if a block is being held
28          forall percept(holding(BlockID)) do insert(
                holding(BlockID)).
29          forall percept(not(holding(BlockID))) do delete(
                holding(BlockID)).
30
31          %remove blocks that are not held or in the room
                anymore
32          forall bel(in(Place), block(BlockID, ColorID,
                Place), not(holding(BlockID))), not(percept(
                color(BlockID, ColorID))) do delete(block(
                BlockID, ColorID, Place)).
33
34          %remove blocks that are not held or in the room
                anymore
35          forall bel(in(Place), block(BlockID, ColorID,
                Place), not(holding(BlockID))), not(percept(
                color(BlockID, ColorID))) do delete(block(
                BlockID, ColorID, Place)).
36
37          % Update sequence when in dropzone.
38          if percept(sequenceIndex(X)), bel(sequenceIndex(
                OldX)) then delete(sequenceIndex(OldX)) +
                insert(sequenceIndex(X)).
39          if bel( in('DropZone'), seqDone(Seq), length(Seq,
                 N), sequenceIndex(X), N < X , sequence(
                NewSeq), length(NewSeq, X), append(NewSeq, _,
                Full) ) then delete( seqDone(Seq) ) + insert(
                seqDone(NewSeq) ).
40
41          %remove obsolete goals
42          if goal(holding(BlockID)), bel(not(block(BlockID,
                 ColorID, Place))) then drop(holding(BlockID))
                .
43
44          %adopt new goals (and stop traveling)
```

```
45          if not(goal(in(Place))), goal(seqDone(_)) then
                adoptgoals.
46  }
```

A.2 Common Code

robot.act2g:

```
1   use robot as knowledge.
2
3   % The goTo action makes the agent move to a place (
        location) in the BW4T environment.
4   % As long as the agent has not arrived at the place it is
        going to, it will be in "traveling" mode.
5   define goTo(Location) with
6           pre { not(state(traveling)) }
7           post { true }
8
9   % goToBlock only when not traveling and in a room
10  define goToBlock(BlockID) with
11      pre {in(_), not(state(traveling)) }
12          post { true }
13
14  % pickUp can only be performed when not holding a block
15  define pickUp with
16          pre{not(state(traveling)), not(holding(_))}
17          post{ true }
18
19  % putDown can only be performed when holding a block
20  define putDown with
21          pre{not(state(traveling)), holding(_)}
22          post{ true }
```

UpdateSequence.mod2g

```
1   use robot as knowledge. order=linearall.
2
3   module updateSequence{
4           %update sequence
5           if bel(seqDone(SDone), nextNeededColor(ColorID),
                append(SDone,[ColorID],NewSDone) ) then delete
                (seqDone(SDone)) + insert(seqDone(NewSDone)).
6
7           %remove beliefs about the delivered block (if
                this agent was delivering it)
```

```
8           if bel(in('DropZone'), holding(BlockID), block(
                BlockID, ColorID, Place)) then delete(block(
                BlockID, ColorID, Place)).
9   }
```

robotGoals.mod2g

```
1   use robot as knowledge. order=linear.
2
3   module adoptgoals{
4           %If holding the next needed block go to the
                dropzone.
5           if bel( holding(BlockID), nextNeededColor(ColorID
                ), block(BlockID, ColorID, _)) then adopt(in('
                DropZone')).
6
7           %Otherswise, If the next needed block is known
                then adopt a goal to go there and hold it.
8           if bel( nextNeededColor(ColorID), block(BlockID,
                ColorID, Place)) then adopt(in(Place),holding(
                BlockID)).
9
10          %Otherwise go to a random room we haven't seen
                yet'.
11          if bel( not(finished), bagof(Place, (room(Place),
                 not(dropZone(Place))), Places), random_member
                (Dest, Places) ) then adopt(in(Dest)).
12  }
```

robotInit.mod2g

```
1   use robot as knowledge.
2
3   module robotInit {
4           % Store map information, i.e., navigation points
                in the agent's belief base.
5           forall percept(zone(ID, Name, X, Y, Neighbours))
                do insert( zone(ID, Name, X, Y, Neighbours) ).
6
7           % Record the initial state of movement in belief
                base.
8           if percept(state(State)) then insert( state(State
                ) ).
9
10          % Record goal sequence
11          if percept( sequence(Seq) ) then insert(sequence(
                Seq), seqDone([])) + adopt(seqDone(Seq)).
```

Agents\chance of failure	reference	cuttoff	5%	10%	25%	50%	95%	distance	50% send failur
Blockbuster Mincom	193	101	109	98	74	64*	63		45*
Blockbuster dzone only	199	11	98	50	15	5	16		
Blockbuster all room	190	18	85	33	13	52	38		60*
Maximum Coordination	188	168	182	182	162	138	64		

Table 3: The amount of successes over 200 runs per agent per failure mode. All agent and failure mode combinations resulted in a chi square $p < 0.01$ except for Coordination at 5%($p \approx 0.07$) The entries marked with * have been performed with fewer than 200 runs and scaled up to the same amount so less accurate.

```
12          if percept( sequenceIndex(X) ) then insert(
                sequenceIndex(X) ).
13
14          % Adopt initial goal going to a random place
15          if bel(room(PlaceID), PlaceID\='DropZone') then
                adopt(in(PlaceID)).
16  }
```

A.3 Exploratory research data