# Any-Com Multi-Robot Path-Planning with Dynamic Teams: Multi-Robot Coordination under Communication Constraints

Michael Otte and Nikolaus Correll

**Abstract** We are interested in finding solutions to the multi-robot path-planning problem that have guarantees on completeness, are robust to communication failure, and incorporate varying team size. In this paper we present an algorithm that addresses the complete multi-robot path-planning problem from two different angles. First, dynamic teams are used to minimize computational complexity per robot and maximize communication bandwidth between team-members. Second, each team is formed into a distributed computer that utilizes surplus communication bandwidth to help achieve better solution quality and to speed-up consensus time. The proposed algorithm is evaluated in three real-world experiments that promote dynamic team formation. In the first experiment, a five mobile robot team plans a set of compatible paths through an office environment while communication quality is disrupted using a tin-can Faraday cage. Results show that the distributed framework of the proposed algorithm drastically speeds-up computation, even when packet loss is as high as 97%. In the second and third experiments, four robots are deployed in a network of three building wings connected by a common room. Results of the latter experiments emphasize a need for dynamic team algorithms that can judiciously choose which subset of the original problem a particular dynamic team should solve.

## 1 Introduction

Autonomous navigation is a basic primitive of autonomous mobile robots and enables a large number of higher-level tasks that are relevant in commercial and consumer settings. In general, the *multi-robot navigation problem* is to find a coordinated set of collision-free paths for all robots moving within a common area. *Centralized* multi-robot navigation algorithms provide the best completeness guarantees of any tool in the multi-robot planning toolbox; however, they are also the most ex-

Michael Otte and Nikolaus Correll
University of Colorado at Boulder, e-mail: {michael.otte, nikolaus.correll}@colorado.edu

pensive to use. In general, multi-robot navigation is an instance of the piano mover's problem—for which complete solutions have been shown to be exponentially difficult to calculate in the number of robots involved (Schwartz and Sharir (1985)). While this result is sobering, our work attempts to push the limits of what complete multi-robot navigation algorithms can achieve. This is important because the only alternative is to use less expensive incomplete algorithms that may fail to find a valid solution when one exists or lead to dead-lock. That said, many incomplete methods have proven to be extremely useful for all but the most challenging navigation instances, and we advocate using them whenever possible. Our work is most applicable to the difficult situations in which less expensive methods fail.

We investigate a probabilistically complete distributed multi-robot navigation algorithm that enables a team of robots to collaboratively solve their mutual navigation problem, while also being robust to the partial communication failure that occurs in real-world environments. Due to real-world environmental constraints on wireless network quality, we are interested in algorithms that exploit the utility of unreliable communication channels, but also take full advantage of high-quality networks. In previous work (Otte and Correll (2010)) we propose the *Any-Com intermediate solution sharing algorithm* (Any-Com ISS). The basic idea is to parallelize a probabilistically complete any-time random-tree algorithm to distribute the computational load among the entire robotic team. Together, the team finds a sub-optimal solution as quickly as possible, then refines that solution subject to both communication and time constraints. The idea is inspired by previous work on any-time algorithms by Boddy and Dean (1989) and Ferguson and Stentz (2006).

As the name suggests, a key element of Any-Com ISS is the sharing of intermediate solutions among team members, where intermediate solutions are the asymptotically optimal solutions found by different robots as time progresses. Robots that receive ISS data are able to prune their local portions of the global search tree to reflect the best solution known to the sending robot (and, by extension, all robots the sender has received messages from, etc.). Aggressive pruning focuses the team's remaining effort on finding even better solutions. Robots receiving ISS messages are also able to directly improve the best solutions known to the sending robots.

This paper extends our previous work on Any-Com ISS by giving it the ability to use dynamic teams similar to those described by (Clark et al (2003a)). Any-Com ISS and dynamic teams compliment each other for a number of reasons. First, Any-Com ISS assumes that robots working on the same problem are within communication range, and dynamic teams can be used to ensure this condition is met. Second, dynamic teams provide a graceful way to handle the discovery of previously unknown robots; once discovered, Any-Com ISS provides a framework to harness their computational power. Third, and most importantly, they are both designed to attack the centralized multi-robot problem—albeit from different angles. By combining the two ideas, we hope to create a more resilient algorithm that inherits the positive aspects of both.

Previous work has focused on the necessity of dynamic teams due to limited communication. In Clark et al (2003a) team formation is seen as a positive event and allowed to occur as soon as two robots can communicate. This is because limited communication necessarily hinders team formation and prevents information exchange. In contrast, we believe aggressive team formation overlooks the algorithm's natural ability to break an $r \geq 2$ robot problem into two separate problems of size $n$ and $m$, where $n+m=r$. Let the size of the workspace be denoted $c$. It is much easier for two teams to solve $O(c^m)$ and $O(c^n)$ complex problems in parallel, than for one large team to solve an $O(c^{n+m})$ problem. Therefore, we believe teams should be kept as small as possible, and only add robots to a team if doing so is necessary to avoid collisions and maintain completeness guarantees.

In Section 4 we perform three experiments. The first evaluates performance of a five robot team using Any-Com ISS in an environment deliberately made hostile to communication with tin-can Faraday cages. Experiments two and three evaluate the performance of Dynamic Team Any-Com ISS when used in Andrews Hall, a large three-wing residence hall on the University of Colorado at Boulder campus. The rest of this paper is organized as follows: related work is presented in Section 2, discussion of results is given in Section 5, and conclusions are drawn in Section 6.

## 2 Related Work

There exists a large body of literature on the multi-robot navigation problem. Previous approaches range from simple reactive solutions to fully coordinated, centralized solutions. In the *cocktail party* model all robots are ignorant of other agents' intentions (Lumelsky and Harinarayan (1997)). Each agent maintains its own goals, world-view, and navigation function. Other robots are viewed as obstacles, and the control loop alternates between sensing, planning, and movement. If all robots use the same algorithm, then the estimated movement of other robots can be refined (van den Berg et al (2009)). Cocktail party algorithms are incomplete (e.g. fail when two robots must move in opposite directions through a narrow corridor), but they are popular due to their simplicity, scalability, and communication free architecture.

Often a set of *traffic rules* is used to facilitate navigation similar to the way automobiles (theoretically) interact via traffic laws (Alami et al (1998); Kato et al (1992); Ryan (2008)). These methods assume each robot knows the rules, agrees to follow them, and can sense required environmental cues (e.g. stoplights). Traffic rules are simple, distributed, and scalable. However, they assume highly structured environments, and may contain rules that prohibit optimal solutions from being found (e.g. taking a short-cut by going the wrong way down a one-way street).

*Prioritized planning* forces robots to respect the movement constraints imposed by higher priority robots (Buckley (1989); Erdmann and Lozano-Perez (1987); Warren (1990)). The highest priority robot plans first, then the next-highest, and so forth. On-line versions exist that alternate sensing, planning, and movement (Clark and Rock (2001); Hada and Takasa (2001); Yeung and Bekey (1987)). Prioritized methods are incomplete—higher priority robots follow optimal or near-optimal trajectories, but lower priority robots may be unable to find a solution. A similar idea has also been used to periodically guarantee line-of-sight communication while performing coverage (Hollinger and Singh, 2010).

*Decoupled planning* works in two phases (Aronov et al (1998); Guo and Parker (2002); Kant and Zuker (1984); Leroy et al (1999); Simeon et al (2002)). Robots calculate their own path to the goal in phase 1, while ignoring all other robots. Space-time positions along these paths are then calculated in phase 2. Priorities may be assigned for the phase 2 calculation (Asarm and Schmidt (1997); Bennewitz et al (2001)), and special cases exist for two robots (Lee and Lee (1987); O'Donnell and Lozano-Perez (1989)). Phase 1 is completely distributable, but phase 2 must be performed on a single agent (or in parallel on each robot). Decoupled planning can be distance-optimal but is incomplete because paths are determined after phase 1, and may systemically conflict (Sanchez and Latombe (2002b)).

In *Centralized planning* all robots are viewed as individual pieces of a single composite robot. Paths are calculated in the resulting high-dimensional configuration space (Bonert et al (2000); Clark et al (2003b); Everett et al (1994); Parsons and Canny (1990); Ramanathan and Alagar (1985); Sanchez and Latombe (2002a,b); Schwartz and Sharir (1985); Xidias and Aspragathos (2008)). The high-dimensional solution is then projected into the relevant subspace for each robot. Centralized planning is theoretically optimal and complete, but practical algorithms are usually probabilistically or resolution complete/optimal. Centralized planning provides the best guarantees of any multi-robot planning method, but it is also the most computationally complex. Dynamic teams have previously been used in conjunction with centralized algorithms by Clark et al (2003a).

With respect to path-planning, previous work in centralized algorithms is the most closely related work to our own, but requires each agent to calculate the entire solution completely on its own. In contrast, our Intermediate Solution Sharing idea leverages the distributed-computing power of the robotic team to help find better solutions more quickly. Non-centralized approaches are incomplete, while Any-Com ISS is probabilistically complete. Previous work has not considered what happens when communication deteriorates, which is a main contribution of our work.

With respect to dynamic team formation, the most closely related work is Clark et al (2003a). In that work, the authors form teams as soon as robots are within communication range. In our work, we delay team creation until it is necessitated by conflicting solutions. By keeping team sizes small, we hope to minimize the complexity of the problems faced by the individual teams. We also perform experiments in a much larger workspace that subjects robots to actual (i.e., not simulated) wireless communication disturbances.

## 3 Methodology

For path planning we use the Any-Com ISS algorithm for multi-robot navigation (Otte and Correll (2010)). This is a modified Any-Time Rapidly Exploring Random Tree (RRT) algorithm with different random seeds per robot (leading to approximately parallel search), and aggressive pruning of local search-trees based on the best intermediate solution known to any robot. The underlying random-tree algorithm is similar to RRT* (Karaman and Frazzoli (2010)) in that it judiciously chooses how to attach new nodes to the tree in order to eliminate the winding and wandering normally associated with RRT.

The ongoing information exchange between robots during the planning phase is also used to jump-start the consensus building that is necessary to ensure all robots in a particular team use the same final solution. Successful communications help to find better solutions and agreements more quickly, while unsuccessful communications do not hinder the eventual discovery of a solution/agreement. In practice, robots must fall-back to using an incomplete method if communication totally fails (e.g., after a period of communication black-out).

In this paper we extend Any-com ISS to incorporate dynamic teams of robots. Messages passed between robots are modified to include:

- List of all members in the sender's team ($s.team$, where the sending robot is $s$)
- Start and goal of the sender's team
- Current planning epoch $s.E_r$ of each robot $r$ in the sender's team
- Current location $s.l_r$ of each robot $r$

In addition to the usual ISS data:

- Best solution $s.sln$ currently known to the sender (for the sender's problem)
- Best solution's length $s.bstln$
- ID of the robot that generated $s.bstln$
- List of robots that have submitted a final solution
- List of robots that support $s.sln$

Robots send messages that reflect their most current view of the global problem state. This means that each robot is responsible for adding itself to the appropriate lists in its outgoing messages. All messages are sent using the User Datagram Protocol (UDP), which is a connection-less protocol that drops messages when communication fails. Dropping old messages helps keep network data current.

Dynamic Team Any-Com ISS is outlined in Figure 1. *ID* is the global identification number of the local robot. Each robot starts in its own team and attempts to find a path to the goal (**DynamicTeamAnyComISS**$()$, lines 1 and 5, respectively). *rst* is a flag used to indicate when the random tree should be restarted due to a change in the planning problem (lines 4, 6). **RandomTree**$()$ attempts to find a solution to the current problem using the algorithm described in Otte and Correll (2010); it returns successfully after the team has found and agreed upon a solution, or unsuccessfully if $rst =$ **true**. The subroutine **FollowPath**$()$ causes a robot to follow its projection of the global solution (line 7) and **StopRobot**$()$ causes a robot to halt (line 9).

Robots periodically broadcast their best known solutions at rate $\omega$ (**Messaging**$()$, line 12). *r.team* and *team* are the sets of teammates of robot $r$ and the local robot, respectively. **Dist**$(l_a, l_b)$ returns the minimum distance between robots $a$ and $b$ in the workspace (line 14). **Conflict**$(a.sln, b.sln)$ returns true if the two solutions $a.sln$ and $b.sln$ conflict (line 14). Teams are combined if their best intermediate or final

```
DynamicTeamAnyComISS()
 1: add $ID$ to $team$
 2: on a separate process do
    Messaging() at rate $\omega$
 3: for forever do
 4:     $rst = $ false
 5:     RandomTree()
 6:     if not $rst$ then
 7:         FollowPath()
 8:     else
 9:         StopRobot()
```

```
Messaging()
 1: if message from $r \in team$ and $r.E_r \geq E_r$ then
 2:     if $r.team \neq team$ then
 3:         for $\forall$ $a$ s.t. $a \in r.team$, $a \notin team$ do
 4:             add $a$ to $team$
 5:         $E_{ID} = E_{ID} + 1$
 6:         $rst = $ true
 7:     for $\forall$ $t \in team$ do
 8:         $E_t = \max(E_t, r.E_t)$
 9:     if $r.bstln < bstln$ then
10:         add $r.sln$ to local search-tree
11:         $bstln = r.bstln$
12:     broadcast messages
13: else if message from $r \notin team$ then
14:     if $ID \in r.team$ or (Conflict($r.sln, sln$) and
        $\exists a \in r.team$, $\exists b \in team$ s.t. Dist($l_a, l_b$) $< \delta$) then
15:         for $\forall$ $t$ s.t. $t \in r.team$, $t \notin team$ do
16:             add $t$ to $team$
17:         $E_{ID} = E_{ID} + 1$
18:         $rst = $ true
```

Fig. 1: Dynamic Team Any-Com ISS (left), and messaging subroutine (right).

solutions interfere with each other and robots creating/following those solutions are closer than a predefined threshold $\delta$ (lines 2-6 and 14-18). $E_r$ is an integer that is incremented every time robot $r$'s problem changes. This is done to ensure outdated messages pertaining to old planning problems are ignored. $r.bestln$ and $bstln$ are the lengths of the best intermediate solutions known to robot $r$ and the local robot, respectively. If a message is received with a better solution, then that solution is added to the local tree (lines 9-11).

There are a number of reasons to delay team formation as long as possible. First, complete multi-robot navigation is exponentially complex in the number of robots, so keeping teams small minimizes the problem complexity and helps to find solutions more quickly. Second, although any-com ISS is theoretically robust to poor communication, it cannot work when communication completely fails. Good communication also leads to improved performance. Assuming communication is generally better at closer range, forming teams such that teammates are near each other reduces the chance of total communication failure.

Even when two teams $A$ and $B$ are in conflict, we delay team combination until a robot in $A$ is closer than $\delta$ to a robot in $B$. This is because there may exist another robot/team $C \not\subset (A \cup B)$ initially beyond communication range, but that eventually joins $A$ or $B$ and forces them to re-plan. Thus, the "conflicting" solution of $A$ or $B$ may either become invalid or cease to be in conflict before either team actually reaches the $A$ vs. $B$ conflict. In contrast, if $A$ and $B$ are in conflict *and* two robots are physically near each other, then the chances of collision are greater, and we combine the teams.

Any-Com ISS is probabilistically complete, and assumes all robots are within communication range of each other. Let $r$ be the minimum distance for robots to detect a conflict and safely stop without colliding, measured between robot centers.

**Theorem 1:** *Dynamic Team Any-Com ISS is probabilistically complete for any $\delta$ such that $r \leq \delta < \infty$, assuming a finite number of robots and all robots are within communication range.*

**Proof:** When robots in conflicting teams become closer than $\delta$, their teams are combined. Teams are never uncombined. Teams continue to plan, move, and combine until solutions are found that do not result in further combination. The latter will happen in two cases:

1. All robots belong to a single team—which is the same as using Any-Com ISS.
2. No conflicting teams ever get closer than $\delta$—so all remaining teams can effectively ignore each other. This is the same as each team individually using Any-Com ISS.

Either (1) or (2) must happen since the number of robots is finite. In either case, the problem is reduced to running Any-Com ISS, which is probabilistically complete. $\square$

## 4 Experiments

We use the Prairiedog robotic platform (Figure 2, right). Robots run the ROS operating system, localize using the Hagisonic Stargazer, and are equipped with a map of the environment (Figure 2-left and Figure 4). Robots exchange data using IEEE 802.11g wireless in ad-hoc mode. The target speed is set to 0.2 meters per second.

### *4.1 Faraday Cage Experiment*

In this experiment five robots plan paths through an office environment while the wireless channel is systematically disturbed by shielding the RF system using a tin-can Faraday cage. We expect the imperfect Faraday cage to significantly disrupt communication—but not completely prohibit it. Statistics are collected on the actual solution quality and consensus time as a function of the effective packet throughput. Solution quality is measured in terms of the maximum time to goal for any robot, and consensus time is defined as the time after planning stops until the entire team agrees on a single solution.

Any-Com ISS is compared to a client-server method—henceforth referred to as *baseline*. While Any-Com ISS uses the entire team for distributed computation, baseline calculates its solution on a single agent and then distributes it to the other robots. Simulations in our previous work have predicted that Any-Com ISS should outperform baseline, even given packet-loss rates as high as 95% (Otte and Correll

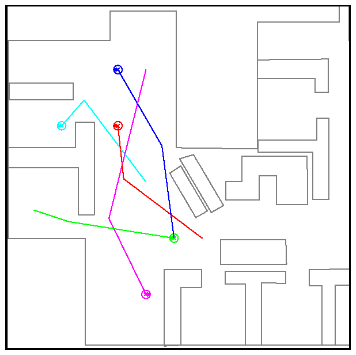
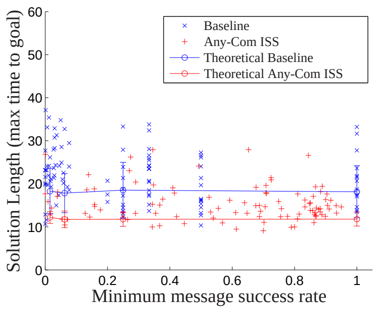


Fig. 2: Left: 5 robots in an office environment and the resulting paths. Right: Prairiedog platform. Each robot is equipped with an indoor localization system, a netbook and IEEE 802.11g wireless communication.



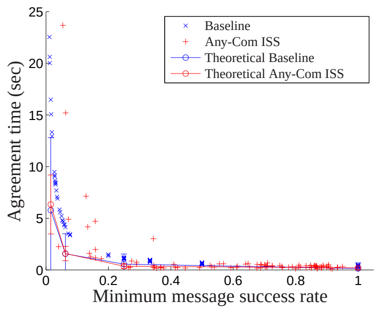


Fig. 3: Faraday cage experiment results. Crosses and stars correspond to the effective, measured packet loss in the environment. Error-bars show means and standard deviations over 40 simulations assuming Poisson distributed package loss.

(2010)). However, the theoretical model used for simulation assumes that communication quality is Poisson distributed, and cannot account for all real-world communication disturbances. This experiment is designed to test Any-Com ISS under harsh communication constraints in a real-world environment, and either validate or refute the earlier simulated predictions.

Experimental results are depicted in Figure 3, along with predictions from the theoretical simulations. The real data are depicted as crosses/stars. The theoretical data are shown as circles and error bars (representing mean and standard deviation over 40 simulations, respectively, per a particular drop-rate). Note, the vertical appearance of the data-points at 1, 1/2, and 1/3 is an artifact of the low number of messages sent by the baseline approach and caused by experiments where solutions were successfully distributed on the first, second, and third, attempts, respectively.

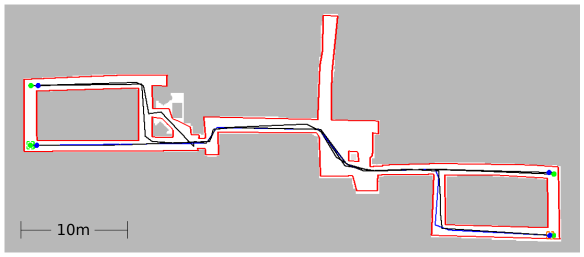


Fig. 4: Andrews Hall (gray and white), with the polygon obstacle map (red), and Conflicting solutions found when robot team sizes = 1 (black paths). Each robot plans from its current location (blue) to a goal location (green).

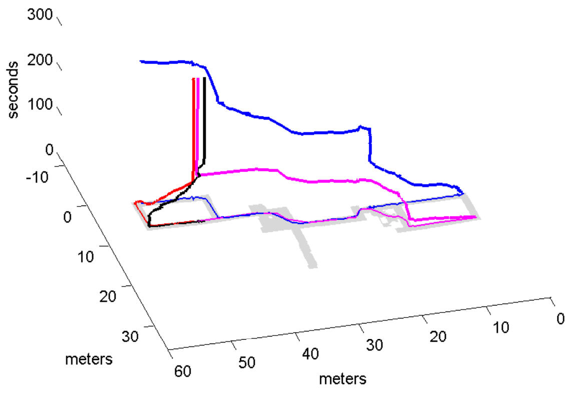


Fig. 5: Actual robot paths vs. time (thick colored lines), and their projections on the free-space of Andrews Hall (thin colored lines and gray, respectively). Time 0 corresponds to the start of movement. Blue successfully reaches the goal, but the experiment is aborted when the red + black + pink team is unable to find a solution after 10 minutes. Note, only the first 5 minutes of the experiment are plotted.

### *4.2 Large Andrews Hall Experiment*

In this experiment we deploy four robots in Andrews Hall, a large building on the University of Colorado at Boulder campus with a challenging floor-plan that emphasizes team formation (see Figure 4). Two robots start in each of the east and west wings of the building, respectively. Robots are assigned the task of trading places with a robot in the opposite wing. As discussed in the previous section, robots initially plan a path to the goal for themselves, then form larger teams and re-plan if they encounter other robots/teams who's paths conflict with their own. Team formation distance $\delta$ is set to 3 meters.

We performed 5 runs of the experiment and observe failure or partial failure in all runs. That is, never did all four robots successfully reach their goals in a single

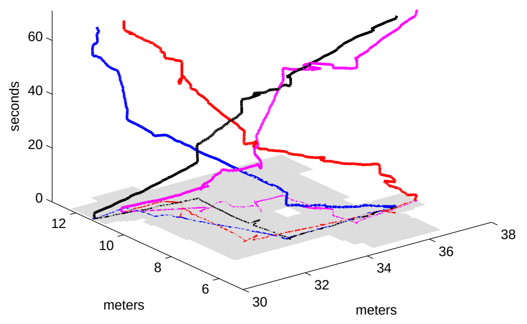


Fig. 6: Workspace is limited to the large common room. Actual robot paths vs. time (thick colored lines), and their projections on the free-space (thin colored lines and gray, respectively). Time 0 corresponds to the start of movement. The resulting robot paths are plotted vs. time (thick colored paths), along with their projections on the free-space (colored paths and gray, respectively).

run. We postpone a full discussion of this result until the next section; however, the failure is due to the computational complexity of a large workspace combined with large dynamic team size. Planning through the entire residence hall is too complex a problem for teams of more than two robots to handle in any reasonable amount of time, and the design of the building makes it unlikely that all robots will reach their goals without forming teams of size three or four. Over all five runs, the average observed communication quality between any two members of the same team is 60.69% with a standard deviation of 22.64%—so communication quality is not to blame for the team's repeated failure to find a solution.

Figure 5 illustrates robot location vs. time from a typical run of the experiment. The red robot catches up to the black robot and they form a dynamic team in the east wing of the building (the left wing of the plot). Before red and black discover a 2-robot solution, the pink robot joins their team. The resulting 3 robot team is unable to find an initial solution after planning for 10 minutes, and the experiment is halted. In this run, blue is the only robot able to successfully reach its goal.

## *4.3 Small Andrews Hall Experiment*

This experiment is similar to the previous one, except that the 4 robots start in locations as they might enter the large common room in the center of the dormitory. From the combined team's point of view, the workspace is limited to include only the common room. Also, $\delta$ is increased to 10 meters so the robots are able to quickly form one large team when their solutions conflict. We perform 10 runs, and observe

Table 1: Small Andrews Hall Experiment Statistics, 10 runs with 4 robots

| | mean | standard deviation |
|---|---|---|
| In-team communication quality | 67.47% | 26.68% |
| Time to first solution (seconds) | 14.24 | 9.55 |
| First solution length (meters) | 46.59 | 4.30 |
| Final solution length (meters) | 38.02 | 3.27 |
| Actual distance traveled (meters) | 44.88 | 7.80 |

that all robots successfully reach their goals in every run. Figure 6 depicts robot locations vs. time for a typical run. The mean and standard deviation of observed statistics over all 10 runs are displayed in Table 1. Solution length is measured as the sum of all individual path lengths. The mean measured distance robots actually traveled is greater than the mean final solution length due to small pose jumps between localization tags and temporary localization error incurred between global localization updates during rotation.

## 5 Discussion of Results

In the Faraday cage experiment, Any-Com ISS out-performs the baseline method in terms of solution quality (i.e., path-time-to-goal). The two methods are similar with respect to agreement time, except that Any-Com has a lower standard deviation—which we attribute to early consensus building during the planning phase. Basic trends in the experimental results are similar to those predicted by the theoretical simulations. However, in practice, we observe that agreement times are greater than those predicted by the theoretical model. We believe this is due to the fact that wireless communication is not Poisson distributed in the real world. We also find that Any-Com ISS can drastically speed up computation, even if packet loss is as high as 97%. In other words the team is able to act as an effective distributed computer with only 3% of the messages getting through.

In the large Andrews Hall experiment we observe that the configuration space of teams involving three or more robots is so large that the problem becomes intractable. This happens despite the distributed computing power of the robotic team using Any-Com ISS, and our efforts to delay team formation (and thus problem complexity) as much as possible. The observation that intra-team packet loss was less than 40%, on average, suggests that the team's difficulty in finding a solution was due to problem complexity and not poor communication. This result is interesting because it shows that dynamic team formation can lead to grid-lock and failure because it forces teams to deal with the worst-case complexity of the problem. In contrast, the small Andrews hall experiment shows that solving a reduced problem is well within the computational power of a 4 robot team. These contrasting results highlight the need for new dynamic team algorithms that limit a team's workspace to the minimum subset of the environment required to solve its communal planning problem.

Assume there are two teams of size $n$ and $m$, respectively, that have conflicting solutions. If we desire to find a complete solution to the combined problem, then a larger combined team must be formed (to ensure conflicting robots are aware of each-others' intentions, and thus avoid each other when a new solution is planned). However, chances are good that the two smaller solutions only conflict in a small subspace of the original workspace. Let the complexity of the subspace in which they conflict be denoted $s$. Note that $s \leq c$, where $c$ is the complexity of the original problem. Thus, solving the sub-problem is only $O((s/c)^{n+m}) \leq 1$ times as hard as solving the original problem, and as $s$ becomes small relative to $c$, the relative difficulty of the sub-problem approaches 0. Therefore, it makes sense to have the team focus on re-planning its solution for only the subset of the original problem that is in conflict. This can be achieved by having each team retain the valid portions of their old solutions, while jointly finding a solution to the combined problem through the conflicting subspace—much as shown in Figure 6. The relevant projections of the combined solution can then be sandwiched between the valid ends of each team's original plan. Individual robots can even work on solving the combined sub-problem en-route to their starting locations for it.

One reason we already use dynamic teams is because they allow robots to solve less complex problems in parallel, graduating to more complex problems only when necessary. It seems reasonable that we extend the existing functionality of dynamic team formation to include tractable sub-problem selection, as well. We hope to explore these ideas in future work. Despite all precautions taken during dynamic team formation, in a worst-case scenario all robots may still end up as part of a single team that must plan through the entire environment. When confronted with this worst-case scenario, teams may be unable to find a reasonable solution, and may still need to fall-back to using an incomplete method.

## 6 Conclusions

We experimentally evaluate a distributed centralized multi-robot path-planning algorithm called Dynamic Team Any-Com ISS. Each robot starts in its own team, and teams are joined if their individual solutions conflict. Combining teams based on path conflicts allows non-conflicting teams to solve problems of reduced complexity in parallel. Within a particular team, distributed computation utilizes the computing potential of all team-members to find better solutions more quickly.

In our first experiment, we intentionally disrupt communication using a tin-can Faraday cage and find that Any-Com ISS functions well despite packet loss rates as high as 97%. This result validates theoretical results obtained from simulation, and is important because it shows that a robotic team can function as an effective distributed computer despite poor communication.

In further experiments we evaluate Dynamic Team Any-Com ISS in a large, complex, indoor environment. When instructed to swap places from one end of the building to the other, robots succeed in forming teams after they discover each other. However, the large hyper-volume of the team's configuration space prohibits teams of three or more robots from finding a solution within any useful amount of time. In contrast, teams of four robots are able to solve the similar but smaller problem of trading places across a common room located in the center of the building—a likely place for individually planned paths from the previous experiment to conflict.

Although computational complexity of centralized algorithms is a known theoretical issue, we show experimentally that it can cause dynamic team formation to trigger grid-lock and failure. This illustrates the importance of algorithms for choosing appropriate sub-problems for dynamic teams to solve. We believe that dynamic teams operating in large environments should attempt to solve the smallest sub-problems necessary to avoid collision. For example, by having the combined team plan only from one side of a conflict region to the other, and leaving navigation to and from that area up to the individual robots or smaller teams. Despite the exponential complexity of complete multi-robot problems, this strategy is advantageous because it decreases the size of the base to which the exponent is raised.

Our work on Dynamic Team Any-Com ISS attempts to extend the size of problems for which a complete solution can be calculated. By delaying team formation as long as possible, dynamic teams attempt to minimize the complexity of the problem required to be solved by any particular team. When a team must be formed, Any-Com ISS divides the effort of computing the members' common solution among all robots that solution will benefit.

## Acknowledgment

The authors would like to thank the residents of Andrew's Hall for allowing us to conduce experiments in their home. We would also like to thank Sam Edwards for helping map the building, Erik Komendera for helping run Experiment 2, and Dustin Reishus for his valuable feedback. Finally, we would like to thank members of the Andrews Hall Robotics Initiative for helping us instrument the building with localization tags.

## References

Alami R, Fleury S, Herrb M, Ingrand F, Robert F (1998) Multi-robot cooperation in the martha project. Robotics and Automation Magazine, IEEE 5:36–47

Aronov B, de Berg M, van der Stappen AF, Svestka P, Vleugels J (1998) Motion planning for multiple robots. In: Proceedings of the fourteenth annual symposium on Computational geometry, Minneapolis, USA, pp 374–382

Asarm K, Schmidt G (1997) Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. In: Proc. IEEE International Conference on Robotics and Automation, pp 3526–3533

Bennewitz M, Burgard W, Thrun S (2001) Optimizing schedules for prioritized path planning of multi-robot systems. In: Proc. IEEE International Conference on Robotics and Automation, pp 271–276

Boddy M, Dean TL (1989) Solving time-dependent planning problems. In: Proc. Eleventh International Joint Conference on Artificial Intelligence, pp 979–984

Bonert M, Shu LH, Benhabib B (2000) Motion planning for multi-robot assembly systems. International Journal of Computer Integrated Manufacturing 13:301–310

Buckley SJ (1989) Fast motion planning for multiple moving robots. In: Proc. IEEE International Conference on Robotics and Automation, pp 322–326

Clark CM, Rock S (2001) Randomized motion planning for groups of non-holonomic robots. In: Proc. International Symposium of Artificial Intelligence, Robotics and Automation in Space

Clark CM, Rock SM, Latombe JC (2003a) Dynamic networks for motion planning in multi-robot space systems. In: Proc. 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS, pp 3621–3631

Clark CM, Rock SM, Latombe JC (2003b) Motion planning for multiple mobile robots using dynamic networks. In: Proc. IEEE International Conference on Robotics and Automation, pp 4222–4227

Erdmann M, Lozano-Perez T (1987) On multiple moving objects. Algorithmica pp 477–521

Everett HR, Gage DW, Gilbreath GA, Laird RT, Smurlo RP (1994) Real-world issues in warehouse navigation. In: Proceedings of the SPIE Conference on Mobile Robots IX, Boston, MA, vol 2352, pp 629–634

Ferguson D, Stentz A (2006) Anytime rrts. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 5369–5375

Guo Y, Parker LD (2002) A distributed and optimal motion planning approach for multiple mobile robots. In: Proc. IEEE International Conference on Robotics and Automation, pp 2612–2619

Hada Y, Takasa K (2001) Multiple mobile robot navigation using the indoor global positioning system (igps). In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Hawaii, United States, pp 1005–1010

Hollinger G, Singh S (2010) Multi-robot coordination with periodic connectivity. In: Proc. IEEE International Conference on Robotics and Automation

Kant K, Zuker SW (1984) Trajectory planning in time-varying environments, 1: Tpp = ppp + vpp. In: Technical Report TR-84-7R, McGill University, Computer vision and Robotics Laboratory, Canada

Karaman S, Frazzoli E (2010) Incremental sampling-based algorithms for optimal motion planning. In: Proc. Robotics: Science and Systems VI

Kato S, Nishiyama S, Takeno J (1992) Coordinating mobile robots by applying traffic rules. In: Proc. IEEE International Conference on Intelligent Robots and Systems, pp 1535–1541

Lee BH, Lee C (1987) A minimum-time trajectory planning method for two robots. IEEE Transactions on Systems, Man and Cybernetics 17:21–32

Leroy S, Laumond JP, Simeon T (1999) Multiple path coordination for mobile robots: a geometric algorithm. In: Proc. International Conference on Artificial Intelligence

Lumelsky VJ, Harinarayan KR (1997) Decentralized motion planning for multiple mobile robots: The cocktail party model. Autonomous Robots 4:121–135

O'Donnell PA, Lozano-Perez T (1989) Deadlock-free and collision-free coordination of two robotic manipulators. In: Proc. IEEE International Conference on Robotics and Automation, Scottsdale, AZ, pp 484–489

Otte M, Correll N (2010) Any-com multi-robot path-planning: Maximizing collaboration for variable bandwidth. In: Proc. 10th International Symposium on Distributed Autonomous Systems

Parsons D, Canny J (1990) A motion planner for multiple mobile robots. In: Proc. IEEE International Conference on Robotics and Automation, vol 1, pp 8–13

Ramanathan G, Alagar VS (1985) Algorithmic motion planning in robotics: coordinated motion of several disks amidst polygonal obstacles. In: Proc. IEEE International Conference on Robotics and Automation, pp 514–522

Ryan M (2008) Exploiting subgraph structure in multi-robot path planning. Journal of Artificial Intelligence Research 31:497–542

Sanchez G, Latombe JC (2002a) On delaying collision checking in prm planning: Application to multi-robot coordination. The international Journal of Robotics Research 21:5–26

Sanchez G, Latombe JC (2002b) Using a prm planner to compare centralized and decoupled planning for multi robot systems. In: Proc. IEEE International Conference on Robotics and Automation

Schwartz JT, Sharir M (1985) On the piano mover's problem iii. coordinating the motion of several independent bodies: the special case of circular bodies amidst polygonal barriers. In: Proc. IEEE International Conference on Robotics and Automation, pp 514–522

Simeon T, Leroy S, Laumond JP (2002) Path coordination for multiple mobile robots: A resolution-complete algorithm. IEEE Transactions on Robotics and Automation 18:42–49

van den Berg J, Guy SJ, Lin M, Dinesh Manocha (2009) Reciprocal n-body collision avoidance. In: Proc. International Symposium on Robotics Research

Warren CW (1990) Multiple robot path coordination using artificial potential fields. In: Proc. of IEEE International Conference on Robotics and Automation, Cincinnati, OH, pp 500–505

Xidias EK, Aspragathos NA (2008) Motion planning for multiple non-holonomic robots: a geometric approach. Robotica 26:525–536

Yeung DY, Bekey GA (1987) A decentralized approach to the motion planning problem for multiple mobile robots. In: Proc. IEEE International Conference on Robotics and Automation, vol 4, pp 1779–1784