To appear in *Proceedings of the International Conference on Robotics and Automation* (ICRA 12), St Paul, Minnesota, May 2012.

On Coordination in Practical Multi-Robot Patrol

Noa Agmon[1], Chien-Liang Fok[2], Yehuda Emaliah[3], Peter Stone[1], Christine Julien[2], and Sriram Vishwanath[2]

***Abstract*— Multi-robot patrol is a fundamental application of multi-robot systems. While much theoretical work exists providing an understanding of the optimal patrol strategy for teams of coordinated homogeneous robots, little work exists on building and evaluating the performance of such systems for real. In this paper, we evaluate the performance of multi-robot patrol in a practical outdoor distributed robotic system, and evaluate the effect of different coordination schemes on the performance of the robotic team. The multi-robot patrol algorithms evaluated vary in the level of robot coordination: no coordination, loose coordination, and tight coordination. In addition, we evaluate versions of these algorithms that distribute state information—either individual state, or entire team state (global-view state). Our experiments show that while tight coordination is theoretically optimal, it is not practical in practice. Instead, uncoordinated patrol performs best in terms of average waypoint visitation frequency, though loosely coordinated patrol that shares only individual state performed best in terms of worst-case frequency. Both are significantly better than a loosely coordinated algorithm based on sharing global-view state. We respond to this discrepancy between theory and practice, caused primarily by robot heterogeneity, by extending the theory to account for such heterogeneity, and find that the new theory accounts for the empirical results.**

I. Introduction

Multi-robot patrol is a fundamental application of multi-robot systems [7], [12], [2], [8], [5], [14] in which a team of mobile robots continually visit a target area (e.g., a continuous 2-D environment, a linear path, or a discrete graph) to monitor some change in the environment's state. In this paper we concentrate on *frequency-based patrol* [7], [8], [12] where the goal is to maximize a given frequency criterion, usually *idleness*, i.e., the time between consecutive visits to a particular point within the patrolled region [12]. We focus on evaluating the worst and average idleness for a team of robots patrolling a cyclic set of waypoints.

Previous work on multi-robot frequency-based patrol primarily concentrated on finding optimal patrol strategies for the robots, and evaluating the solutions using theoretical tools like approximation ratio [7], and in simulation [12]. Generally, they prove that a team of homogeneous robots must be tightly coordinated to maintain uniform temporal separation, which guarantees minimal uniform idleness in a cyclic patrol environment. Whether such tight coordination is practical in reality remains, to our knowledge, an open question.

We present an empirical study of multi-robot patrol using a team of mobile robots provided by the Pharos testbed [10].[1] Pharos achieves robot navigation using consumer-grade GPS/compass and communication through ad hoc WiFi. This is due to practical objectives, i.e., low cost and easy infrastructure-less deployment. Using Pharos, we evaluate different coordination mechanisms for maintaining optimal idleness at given waypoints along a cyclic patrol route. Three different coordination mechanisms are examined: *Uncoordinated*, in which the robots do not coordinate their behaviors with their peers (i.e., they act like individual robots instead of a team). *Tightly coordinated*, in which a robot in the team will not continue to move before knowing that all its teammates are synchronized in terms of reaching their respective waypoints, and *Loosely coordinated*, in which each robot is coordinated only with the subset of robots that are within its communication range. Two message update schemes between the robots are used: *individual status messages*—ISM, where each robot sends its peers only its own status; and *global status messages*—GSM, where each robot sends to its peers its entire world view in terms of team status. We examine the influence of using these two schemes on the team's performance in the two coordinated scenarios (tightly coordinated and loosely coordinated).

Experiments with teams of two and three robots are conducted and the idleness at each waypoint is recorded. Results show that loose coordination using ISM performed better than loose coordination using GSM, since it involves each robot waiting for fewer other robots. Tight coordination always fails; the communication scheme is irrelevant when the route forces the robots to have connectivity disconnections, which is expected and common in real-world networks of autonomous robots. Surprisingly, the uncoordinated mechanism performed better in terms of average idleness, however the loosely coordinated algorithm using ISM outperformed the uncoordinated patrol in terms of worst-case idleness.

We respond to the discrepancy between theory and practice, caused primarily by robot heterogeneity, by extending the theory to account for such heterogeneity. Specifically, we provide a theoretical analysis of the average and worst-case idleness of a team of two robots, given that the robots do not have the same velocity, and show that in some cases the uncoordinated behavior outperforms the coordinated one, as

[1]N. Agmon and P. Stone are with the Department of Computer Science, The University of Texas at Austin {agmon,pstone}@cs.utexas.edu

[2]C. Fok, C. Julien, and S. Vishwanath are with the Department of Electrical and Computer Engineering, The University of Texas at Austin {liangfok,c.julien,theory}@mail.utexas.edu

[3]Y. Elmaliah is with the Department of Computer Science, College of Management Academic Studies, Israel elmaliahy@colman.ac.il

[1]Supplemental data, including videos and code documentation, can be found in: http://pharos.ece.utexas.edu/wiki/index.php/ICRA12

observed in the experiments.

II. The Patrol Problem and Coordination Mechanisms

In the problem of frequency-based multi-robot patrol, a team of robots is required to repeatedly visit a set of interesting locations to optimize some visitation frequency criterion. A key metric for evaluating the performance of any solution to this problem is the *idleness* criterion [12], which is the time between visits by any robot to a point of interest.

Theoretically, the optimal algorithm requires that the robots be tightly coordinated, especially in cyclic patrol routes (e.g. [2], [12], [9], [8]). Tight coordination means all robots travel along the cyclic path while always maintaining uniform inter-robot distance (in time). Specifically, if the travel time along the cyclic path is N time units, then given k homogeneous robots, the distance between them should be maintained as N/k throughout the patrol execution.

In reality, constraints posed by the robots (for example the range of their wireless communication interfaces), the environment (rough terrains or large environments in which direct communication between robots is not possible) makes it at times impossible to obtain perfect coordination. Additionally, even robots that consist of the same hardware do not necessarily act uniformly. For example, their velocities might not be the same due to slight differences in calibration and maintenance, and sensor performance.

In this work, we examine the use of the following three coordination mechanisms, and determine empirically and theoretically the best coordination mechanism for a team of k robots patrolling along a cyclic set of waypoints.

Uncoordinated: Each one of the k robots patrols along the set of waypoints oblivious to the presence of the other robots, i.e., the robots act as k teams, each consisting of one robot, and not as a team of k robots.

Tightly Coordinated: The robots are given a list of teammates in advance. Each robot progresses through one waypoint at a time. Upon arriving at a waypoint, it will not continue to move until it is sure that its teammates have also reached their current destination waypoints.

Loosely coordinated: The robots are given a list of teammates in advance, but they also maintain a dynamic list of “active” teammates. If a robot does not hear from a teammate within some time duration (we set this value to be 5 seconds), it will remove the teammate from the list and mark it as “out of range.” Once it hears from a teammate previously noted as “out of range”, it adds the teammate back again to the list of active teammates. Each robot acts in tight coordination only with team members in its active list.

We used two types of message update schemes between the robots:

Individual Status Messages (ISM): Each robot sends its own status to its peers.

Global Status Messages (GSM): Each robot sends its own status plus its known status of its teammates to its peers.

The only means of coordination between the robots is through explicit communication (specifically, using WiFi). They are not equipped with sensors to visually or otherwise identify their teammates. The robots determine their location and headings using GPS and compass. The control of the robots’ actions is done using behavior-based control [15], where the experiment consists of a sequence of one type of behavior: `GoToWaypoint`.

III. Implementation

We implement the multi-robot patrol application using the Pharos testbed [10], which consists of numerous Proteus mobile robots. These robots are highly modular, enabling easy customization for specific applications. For our robot patrol implementation, the robot configuration is shown in Figure 1. It consists of a customized Traxxas Stampede mobile chassis and a plane containing computational elements, a Garmin eTrex GPS receiver, and a CMPS03 digital compass. The Traxxas Stampede provides non-holonomic car-like movement. It is upgraded with a more powerful V11 Rock Crawler Professional motor, a Devantech 20A MD03 motor controller, stiffer suspension springs, an E4 miniature optical wheel encoder, and dual Tenergy Lithium polymer 14.8V 11Ah battery packs. These upgrades enable the robot to easily move and power the computational and sensing elements that reside above the chassis. While its theoretical top speed is 10m/s, we software limit the top speed to 3m/s. When moving at 0.5m/s, its minimum turning radius is approximately 3m. The batteries enable continuous operation of the robot for over six hours.

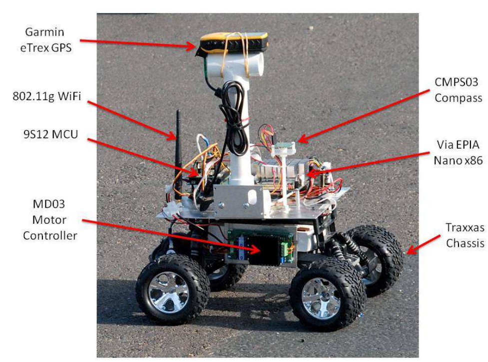


Fig. 1. *One of the Proteus robots used in our tests.*

The robot’s main computational components are a general-purpose x86 computer and a microcontroller. The x86 is a VIA EPIA Nano-ITX motherboard that contains a 32-bit VIA C7 CPU running at 1GHz, 1GB of DDR2 RAM, a 16GB compact flash drive, and a CM9-GP IEEE 802.11g WiFi mini-PCI module based on the Atheros AR5213A chipset. It runs Ubuntu Linux 11.04 server and Player 3.0.2 [11]. Custom Player drivers provide high-level programming abstractions for robot movement and sensing. The WiFi interface is configured to form a wireless ad hoc network on channel 1. This is necessary because as the robots move, they will often go out of range of a central base station, rendering an infrastructure-based network infeasible.

Specialized tasks are handled by a MC9S12DP512 microcontroller (MCU) that is attached to the x86 via a 115.2kbps serial UART. The MCU is responsible for controlling and accessing most of the robot's actuators and sensors. Specifically, it controls the MD03 motor controller via a PWM signal and adjusts the robot's speed based on feedback from the wheel encoder. It also controls the front servo via another PWM signal, which determines the steering of the robot. Finally, the MCU communicates with the CMPS03 digital compass via I2C, gathering heading information at 5Hz and sending this information to the x86.

Localization is achieved using the Garmin eTrex GPS receiver. It is attached directly to the x86 via a 4.8kbps serial UART connection, and provides location information at 0.5-1Hz with approximately 3m accuracy. The GPS receiver is mounted 30cm above the robot on a plastic pole to improve the accuracy and reliability of the device.

As previously mentioned, the x86 runs Player 3.0.2, which provides higher-level software abstractions for the robot's sensors and actuators. Player's `Position2DInterface` is used to control the robot's movements and receive compass data, while the `GPSInterface` is used to receive location data. Finally, an `OpaqueInterface` is used to transfer debugging and MCU status information to the client application, which in this case is the `MultiRobotPatrolServer`.

The `MultiRobotPatrolServer` is written in Java as a Player client. A unique instance of this application runs on each robot and subscribes to the aforementioned Player interfaces. It implements the multi-robot patrol logic. The server uses Java sockets for network communication with other robots. Currently, the robots communicate using singlecast TCP links that are opportunistically formed when two robots are within range. The `MultiRobotPatrolServer` also accepts connections from the `MultiRobotPatrolClient`. The `MultiRobotPatrolClient` runs on a central base station and is responsible for initializing a multi-robot patrol experiment. Initially, all patrol robots are placed within range of the base station. The patrol begins when the `MultiRobotPatrolClient` connects to each robot's `MultiRobotPatrolServer`, and informs it of the team members, patrol type, patrol route, patrol speed, number of patrol rounds, and starting points. The `MultiRobotPatrolClient` then sends a start message to each robot causing it to begin patrolling. Note that by this time, the robots operate independently of the `MultiRobotPatrolClient`. This is necessary since the robots may move out of range of this client while patrolling.

GPS waypoints specify the patrol route. The `MultiRobotPatrolServer` follows this patrol route by encoding it via a behavior-based programming model. Each waypoint in the patrol route is stored within a unique `GoToWaypoint` behavior, which when executed uses GPS and compass data to navigate the robot from its current location to the next waypoint. Thus, the actual patrol activity is implemented as a sequence of `GoToWaypoint` behaviors. Per the behavior-based programming model, behaviors have starting and terminating conditions. In this case, the start condition for the first `GoToWaypoint` behavior is the arrival at the robot's starting waypoint. For the remaining `GoToWaypoint` behaviors, the starting condition is the completion of the previous `GoToWaypoint` behavior. The terminating condition of the `GoToWaypoint` behavior is when the robot reaches the behavior's waypoint and, depending on the type of patrol, synchronizes with the other members of the team.

IV. Evaluation

A. Experimental Settings

To test the multi-robot patrol implementation, we created a patrol route on a parking lot as shown in Figure 2. Waypoints are indicated by the markers, and the route is shown by the arrows. The route consists of six waypoints arranged in a rectangle. The waypoints are spaced 18.5m to 19.5m apart forming a route 114.4m in length per round. In all experiments, the robots patrol the route ten times. We performed experiments involving two and three robots. The robots patrol the route in the same direction (i.e., clockwise when viewed from above). To minimize waypoint idleness, the robots initially start at different waypoints equally spaced along the route. Specifically, for two robot experiments the starting waypoints are 2 and 5, and for three robot experiments the starting waypoints are 1, 3, and 5. For all experiments, the robots move at a target speed of 2m/s, though the actual speed is less because the robots slow when they turn, stop if they fail to receive GPS or compass readings, and stop when synchronizing with teammates in coordinated patrols.

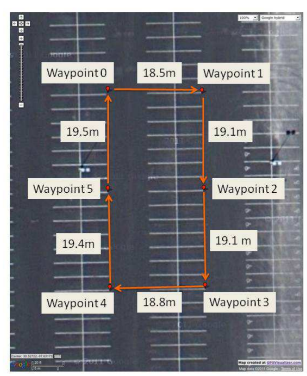


Fig. 2. *The patrol route used to evaluate our multi-robot patrol implementation.*

For two robot experiments, we performed uncoordinated, loosely coordinated, and tightly coordinated tests. Two-robot teams only use ISM messaging since there is no global information to share among the robots other than the status of themselves. For three robot experiments, we perform uncoordinated, loosely coordinated ISM, loosely coordinate

GSM, tightly coordinated ISM, and tightly coordinated GSM.

Each experiment is run twice, enabling us to verify the stability of results across experiments. We observed that the general trend was maintained across different types of experiments and thus, for brevity, we present average values across both uncoordinated and loosely coordinated experiments. The results of the tightly coordinated experiments are omitted because, as will be discussed, they promptly fail due to wireless disconnection between the robots.

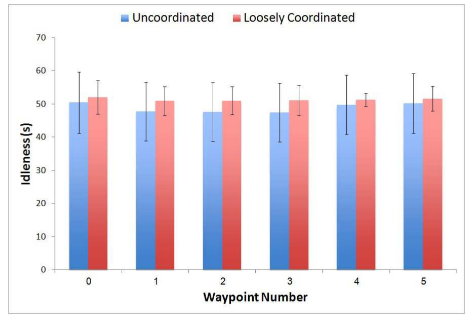
B. Experimental Results

Experiments are successful if they execute to completion. This occurs when all robots complete the desired number of patrol rounds, which is ten in this case. As previously mentioned, each experiment is executed twice. Table I shows which types of experiments executed successfully and which failed. The failed experiments involve tight robot coordination. In these experiments, wireless disconnection resulted in the entire system entering a deadlock state with every robot waiting for another robot to announce their arrival at their waypoint. In a ISM test, a permanent wireless disconnection between any two pairs of robots will result in this deadlock. In a GSM test, any partition among the robots in the network will result in deadlock. The results clearly indicate that in the real-world where wireless links are dynamic and unreliable, an uncoordinated or loosely coordinated strategy is superior because it enables the system to avoid deadlock, i.e., it is more robust to communication failures.

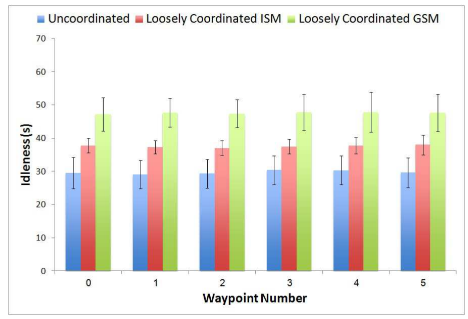
	Two Robots	Three Robots
Uncoordinated	Success	Success
Loosely Coordinated ISM	Success	Success
Loosely Coordinated GSM	N/A	Success
Tightly Coordinated ISM	Fail	Fail
Tightly Coordinated GSM	N/A	Fail

TABLE I

THE SUCCESS AND FAILURE OF MULTI-ROBOT PATROL EXPERIMENTS.



(a) *Two Robots*



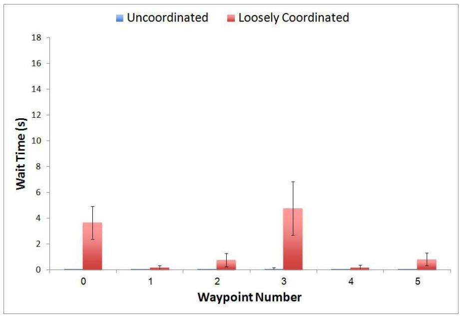
(b) *Three Robots*

Fig. 3. *The average idleness of each waypoint.*

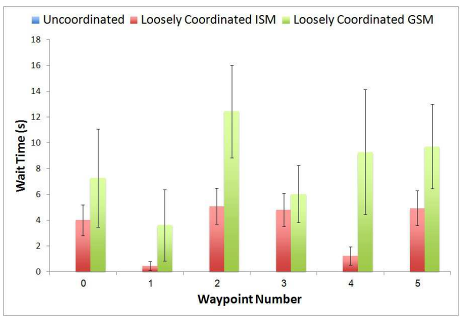
The average waypoint idleness for the successful experiments is shown in Figure 3. The results are the averages across two executions of each the same experiment. Error bars denote 95% confidence intervals. As expected, adding a third robot reduces the overall average waypoint idleness. In the two-robot scenarios, the average of the uncoordinated case is lowest, but its confidence interval is very large relative to the coordinated case. This indicates that loosely coordinating the robots increases the predictability of the waypoint idleness, though at the cost of slightly higher average idleness. In the three robot case, the average idleness in the uncoordinated scenario is far lower than any coordinated scenario, even when considering the confidence intervals. The loosely coordinated ISM scenario performs better than the GSM scenario, despite GSM disseminating more robots status information. The surprising results are caused by coordination overhead, and indicate that any form of coordination will only *increase* the average waypoint idleness, which is undesirable.

To understand why coordinated patrol has increased idleness, consider the amount of time robots spend waiting at a waypoint for their teammates to synchronize. This is shown in Figure 4. Clearly increasing the number of robots results in an increase in the wait time at each waypoint. When GSM coordination is used, the wait time increases even more since robots may wait for teammates who are not directly within wireless range. As expected, the uncoordinated scenario has the least wait time (nearly 0) since the robots do not wait for their teammates at each waypoint. These significant periods where a robot waits for its teammates explain the elevated waypoint idleness in the coordinated scenarios. One interesting observation is that while increasing the number of robots results in a decrease in waypoint idleness, it results in an increase in average wait time at each waypoint since there are more robots with which to synchronize.

The amount of time to complete each experiment is shown in Figure 5. Again, the results are the averages across two executions of each experiment type. In all experiments the robots leave from the same spot, which is our home base next to the patrol route. The start and end times of an experiment are calculated as the time they were launched to the time they reach the final waypoint in the route, respectively. The results show that the three robot uncoordinated scenario is the fastest and that multi-robot coordination increases the total duration. The results also show that GSM coordination in the three robot scenario increases the duration over the ISM case by nearly 300s (5 min.), demonstrating the high overhead of multi-hop coordination. As expected, the uncoordinated



(a) *Two Robots*



(b) *Three Robots*

Fig. 4. *The average wait time at each waypoint.*

scenarios exhibit the highest variation in robot completion times since the robots do not wait for each other and may operate at different speeds due to variations in the calibration of their mobility components and accuracy of the sensors used for navigation.

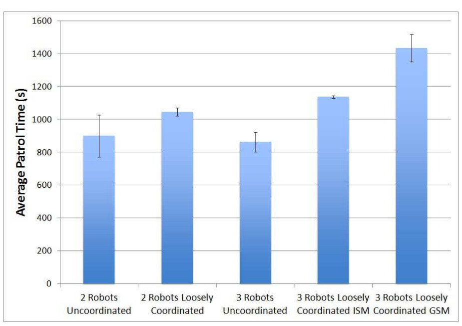


Fig. 5. *The average time to patrol the route ten times.*

In the coordinated scenarios, the robots transmit their status to their teammates approximately once every second. Since the wireless network is opportunistic and dynamic due to node mobility, some of these messages are lost. The number of messages that were transmitted, received, and the percentage lost are shown in Table II. Three robot networks have higher message loss than two robot networks, perhaps due to increased wireless collisions when there are three transmitters. In general, the frequent message loss, among other considerations such as heterogeneity in the robots' velocity, further motivates the use of loose coordination over tight coordination.

	Num. Tx	Num. Rx	Msg. Loss
2 Robots Loosely Coordinated	4,315	3,690	10.8%
3 Robots Loosely Coordinated ISM	13,372	9,042	32.4%
3 Robots Loosely Coordinated GSM	16,768	13,379	20.2%

TABLE II

THE NUMBER OF COORDINATION MESSAGES TRANSMITTED, RECEIVED, AND THE PERCENTAGE OF THESE MESSAGES LOST.

V. OPTIMALITY OF PATROL FOR TWO HETEROGENOUS ROBOTS

As stated in Section IV, a somewhat unexpected experimental result was that the uncoordinated patrol performed better than the coordinated patrol in terms of average idleness. Based on the difference between the patrol time of individual robots (see Figure 5), we concluded that this discrepancy was based on the fact that the robots are not homogenous, but have different velocities. In this section we provide theoretical results that shed light on the empirical superiority of the uncoordinated patrol.

When the robots are homogenous, and specifically have the same velocity, the optimal patrol scheme for a team of k robots was shown to be the tightly coordinated one [4], [7], [8]: The robots should spread uniformly along the cyclic path, and maintain uniform distribution between them throughout the execution. If the time it takes a robot to travel along the entire cyclic path is N time units, then, by maintaining uniform distance (in time) along the path, the guaranteed idleness along the path is exactly N/k. Specifically, the worst idleness along the path and the average idleness at each point along the path is also N/k.

When the robots are not homogenous, and have different velocities, the optimality of the tightly coordinated patrol scheme is not maintained. As mentioned above, we evaluate the patrol using two criteria: *worst idleness*, which is the greatest idleness reported at some point along the path during the execution of the patrol algorithm, and *average idleness*, which is the average idleness at each point along the path.

Denote the velocity of robot R_i, $1 \leq i \leq k$, by v_i (measured, w.l.o.g. in m/sec), and the length of the path by S (measured, w.l.o.g., in $meters$). Assume without loss of generality, that for a 2-robot team, $v_2 > v_1$. Therefore, if the robots are coordinated, then both travel at velocity v_1.

We omit the proofs of the lemmas due to space limitations.

Lemma 1: In a team of two robots, if $2v_1 \leq v_2$, then the coordinated patrol does not perform better than the uncoordinated patrol in terms of both worst *and* average idleness.

Lemma 1 provides only a tight bound on the minimal velocity of R_2 that makes it worthwhile to choose the uncoordinated patrol in terms of *worst idleness*. The average idleness can be lower using the uncoordinated scheme also for lower velocities of v_2. The following lemma provides a tighter bound on the *average* idleness obtained by the coordinated vs. the uncoordinated patrol when $v_1 < v_2 < 2v_1$. Denote the least common multiple of S/v_1 and S/v_2 by $d_t = lcm(S/v_1, S/v_2)$.

Lemma 2: In a team of two robots, if $d_t S/v_1 \leq \frac{d_t}{S/v_2} - 1$, then coordinated patrol does not perform better than the uncoordinated patrol in terms of average idleness.

Returning to the experimental results, we see that in the two-robot uncoordinated patrol, one robot (R_1) completed a pass through the points at an average of approximately 100 seconds, while the second robot (R_2) completed a cycle in an average of approximately 80 seconds. d_t in this case is 400, and hence $\frac{400}{100} = 4 \leq \frac{400}{80} - 1 = 5 - 1 = 4$, and the dominance of the coordinated patrol is not guaranteed. Note that Lemma 2 does not take into account time spent on coordination (exchanging messages to make sure that all robots are synchronized). Therefore, considering also the coordination time, the coordinated patrol is expected to perform even worse than the uncoordinated patrol, as demonstrated in the experiment.

VI. Related Work

The problem of multi-robot patrol has become a canonical problem in multi-robot systems in the past few years due to its immediate applicability in various domains, mainly security and safety applications. As such, there is a great deal of work in this area. Two aspects of the problem are considered in the literature: *adversarial patrol*, in which the team's goal is to maximize its chances of detecting an adversary trying to penetrate through the patrol path undetected (e.g. [5], [2]); and *frequency based patrol*, in which the team's goal is to optimize some frequency criteria, for example to minimize time intervals between visits or to guarantee minimal deviation between such time intervals along the patrol path (e.g. [4], [16], [7], [8]). The latter is the focus of our paper.

The first analysis of the multi-robot patrol problem was presented by Mechado *et al.*[12], where they introduced the notion of *idleness* as the time between consecutive visits along points of interest in a graph. They evaluated several strategies for generating multi-robot patrol paths in simulation. The question they examined was *how to find patrol paths for robots such that the worst idleness is minimized*. Their work did not consider real robotic communication capabilities, and assumed centralized decision making.

Other theoretical work in multi-robot patrol concentrated on similar questions, and varied in their suggested solutions. For example, Elmaliah *et al.*[8] considered multi-robot patrol in areas in which some restrictions exist on the domain (mainly the possibility of representing the world on specific types of grids), finding a minimal cost Hamiltonian cycle in polynomial time while accounting for possible directional velocity constraints. Recently, Portugal and Rocha [16] considered the problem of multi-robot patrol in graphs, where the graph is partitioned such that each robot is assigned to a region of the graph. They evaluated the solution in simulations. Agmon *et al.*[3] described the general problem of multi-robot patrol, accounting for different strategies for dividing a graph between robots—either with or without an overlap between the subgraphs associated with each robot. They evaluated their heuristic solution on a ship simulator, accounting for real environmental and ship constraints, such as sea currents, winds and physical capabilities of the ship. However, their algorithm is also centrally executed, and concentrates on finding an optimal patrol path for the robots.

Elmaliah *et al.*[9] considered the problem of multi-robot patrol along an open fence. In their work they generated an algorithm for determining the optimal patrol path for each robot while accounting for accumulation of errors in robot motion, representing this error in travel time. They performed an evaluation using real robots, and have shown that the algorithm that takes into account these deviations from flawless motion indeed guarantees better point-visit frequencies. However, in their work they also base their patrol on tightly coordinated behavior, where in most parts of the patrol (aside for one endpoint of the open fence) the robots are committed to maintaining uniform distance between them.

Marino *et al.*[13], [14] also considered a real robotic environment, and suggested a fully distributed behavior-based framework for multi-robot perimeter patrol. They do not use explicit communication between the robots, but each robot base its understanding of the world on its sensing capabilities, i.e., by observation. In our work we examine the influence of the communication on the coordination mechanism of the robots, comparing the resulted frequency of visits of a point across the different mechanisms.

Finally, since mobile wireless ad hoc networks are inherently less reliable than traditional networks, many network protocols and middleware services exist that aim to improve reliability in such networks. They touch multiple levels of the network stack and employ diverse strategies including increasing information spread [20], [6], better broadcast scheduling [19], network coding [21], introducing delay-tolerance [1], and context-aware adaptation [18], [17]. In this paper, we utilize the standards-based network stack available on Ubuntu Linux. The aforementioned protocols and services were not used because of their proprietary nature. Regardless, determining their impact on multi-robot patrol is an interesting direction for future work.

VII. Conclusions and Future Work

Due to modeling and approximations, simulations and theoretical results often produce conclusions that do not perfectly reflect reality. In the case of multi-robot frequency-based patrol that we examine in this paper, there is significant inconsistency between theoretically proven optimal behavior and practical results. This divergence originates from a combination of two real-world phenomena: unreliable wireless network connectivity among mobile robots and the heterogeneity of robots in a team. Through experimentation in a real-world robotic testbed, we demonstrated that the uncoordinated and loosely coordinated-ISM algorithms performed better than the theoretically ideal algorithm, which repeatedly fails. Moreover, by relaxing the prior theoretical assumptions to more closely reflect reality, we have shown theoretically that in heterogeneous teams, it is sometimes optimal to act without coordination, as if each robot is part of its own individual team.

As future work, we would like to extend both the theoretical analysis and the empirical evaluation to larger teams of robots. We intend to examine continuous synchronization, where robots do not necessarily stop when waiting for their teammates and can instead continuously synchronize even when between waypoints. We would also like to evaluate how the properties of the mobility plane impacts the performance of the multi-robot patrol algorithm. For example, we could switch to Segway RMP 50-based robots, which can turn in place, but can only move at 1.7m/s (versus the Traxxas Stampede, which can move at up to 10m/s).

VIII. Acknowledgements

This work was funded in part by AFOSR DURIP project number FA9550-07-1-0502. The work was also taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).

References

[1] S. B. A., Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *Communications Magazine, IEEE*, 41(6):128 – 136, june 2003.

[2] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *Proc. of ICRA'08*, 2008.

[3] N. Agmon, D. Urieli, and P. Stone. Multiagent patrol generalized to complex environmental conditions. In *Proc. of AAAI'11*, 2011.

[4] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyr. Recent advances on multi-agent patrolling. *Lecture Notes in Computer Science*, 3171:474–483, 2004.

[5] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proc. of AAMAS'09*, 2009.

[6] E. Biagioni and S. H. Chen. A reliability layer for ad-hoc wireless sensor network routing. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9 - Volume 9*, 2004.

[7] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proc. of IAT'04*, 2004.

[8] Y. Elmaliach, N. Agmon, and G. A. Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Math and Artificial Intelligence journal (AMAI)*, 57(3—4):293—320, 2009.

[9] Y. Elmaliach, A. Shiloni, and G. Kaminka. A realistic model of frequency-based multi-robot fence patrolling. In *Proc. of AAMAS'08*, pages 63–70, 2008.

[10] C. Fok, A. Petz, D. Stovall, N. Paine, C. Julien, and S. Vishwanath. Pharos: A testbed for mobile cyber-physical systems. Technical Report TR-ARiSE-2011-001, University of Texas at Austin, January 2011.

[11] B. P. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proc. of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[12] A. Machado, G. Ramalho, J. Zucker, and A. Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *Proc. of MABS'02*, pages 155–170, 2003.

[13] A. Marino, L. Parker, G. Antonelli, and F. Caccavale. Behavioral control for multi-robot perimeter patrol: A finite state automata approach. In *ICRA*, 2009.

[14] A. Marino, L. E. Parker, G. Antonelli, F. Caccavale, and S. Chiaverini. A fault-tolerant modular control approach to multi-robot perimeter patrol. In *ICRA*, 2009.

[15] L. E. Parker. On the design of behavior-based multi-robot teams. *Journal of Advanced Robotics*, 10:547–578, 1996.

[16] D. Portugal and R. Rocha. Msp algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning. In *Proc of the 2010 ACM Symposium on Applied Computing*, SAC '10, 2010.

[17] G. C. Roman, C. Julien, and H. Qingfeng. Network abstractions for context-aware mobile computing. In *Proceedings of the 24rd International Conference on Software Engineering (ICSE)*, pages 363 –373, 2002.

[18] R. Sen, R. Handorean, G. Roman, G. Hackmann, and C. D. Gill. Knowledge-driven interactions across mobile ad hoc networks. *Int. J. Cooperative Inf. Syst.*, 16(1):123–153, 2007.

[19] M. Slavik and I. Mahgoub. Statistical broadcast protocol design for unreliable channels in wireless ad-hoc networks. In *Global Telecommunications Conference (GLOBECOM)*, pages 1 –5, 2010.

[20] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Efficient routing in intermittently connected mobile networks: The multiple-copy case. *IEEE/ACM Transactions on Networking*, 16(1):77 –90, 2008.

[21] Z. Yang, M. Li, and W. Lou. R-code: Network coding based reliable broadcast in wireless mesh networks with unreliable links. In *Global Telecommunications Conference (GLOBECOM)*, pages 1 –6, 2009.