2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

A Practical Implementation of Random Peer-to-Peer Communication for a Multiple-Robot System

Chris A. C. Parker
Department of
Computing Science
University of Alberta
Edmonton, Alberta, Canada
Email: parkerca@ieee.org

Hong Zhang
Department of
Computing Science
University of Alberta
Edmonton, Alberta, Canada
Email: zhang@cs.ualberta.ca

Abstract— **In this paper, we present a physical implementation of Random Peer-to-Peer (RP2P) communication for use in a multiple-robot system and analyze its performance. Traditionally, multiple-robot systems have either broadcast all of their inter-robot communication or have avoided explicit communication altogether. RP2P communication, on the other hand, allows efficient system-level communication while retaining the error-correction capabilities of peer-to-peer connections. We demonstrate that RP2P communication can be implemented with off-the-shelf components. MRS as large as ten robots are investigated and it is demonstrated that message rates as high as 50 messages/second are easily achievable using TCP connections and 802.11B wireless network interfaces.**

I. Introduction

The robots that compose a multiple-robot system (MRS) must explicitly communicate with each other if they are to demonstrate truly cooperative behavior [1]. Without *explicit* communication, individual robots are limited to inferring the internal states of their teammates through passive observation. This latter form of communication is known as *implicit* communication. Traditionally, inter-robot communication has been implemented explicitly using broadcast communications or implicitly via stigmergic [2] interaction.

A message that is broadcast is received by all robots within range of the sender, regardless of its relevance to them. Thus the application layer of every robot within a system must process every message that is received in order to determine whether the message was intended for it or not, as it is the actual content of a message that would determine its relevance. Further, error correction is not practical when messages are broadcast, as the multiple recipients have no means of coordinating their acknowledgments back to the sender. Because of this, broadcast communication at best is limited to short messages that are not mission-critical, as their delivery intact - or at all - can not be assured. In centralized MRS, a central control robot conceivably could control the access to the broadcast channel, but what about decentralized MRS? In a decentralized MRS [3], no robot is in charge. With multiple robots competing for access to the channel, data collision is inevitable without some scheme for sharing the wireless spectrum. Further, because messages could originate from any member of the system, the number of messages transmitted per unit of time would increase with system population size, placing an ever increasing demand on the robots' application layers.

We are interested primarily in explicit communication within a decentralized MRS. Explicit inter-robot communication in decentralized MRS is difficult to implement since the behavior of these systems emerges from the local interactions of the individual robots (bottom-up) rather than from a single leader's commands (top-down). Our previous work introduced a communication scheme to implement communication in a decentralized MRS along with a characterization of its behavior based upon a mathematical and simulation-based analysis [4]. We call this scheme Random Peer-to-Peer, or RP2P. In this paper, we present a study of RP2P's performance based on physical experiments that were carried out using multiple physical robots and commonly available 802.11B wireless network interfaces. We are not concerned with *what* is communicated between robots. Rather, our goal is to verify that, under some realistic assumptions, RP2P with negligible delay is realizable in a real-world environment with easily obtainable, off-the-shelf components. We further derive the characteristics of RP2P as network contention (e.g., MRS population size and message transmission rate) increases.

The remainder of this paper is laid out as follows. In the next section, we present an overview of RP2P and summarize the aspects of its behavior that we experimentally verify in this work. In Sections III and IV, we describe the hardware and software that were used to conduct our experiments as well as describe how the data from our experiments is analyzed to produce our results. Section V presents our immediate experimental results and Section VI provides a discussion of these results along with some of the broader issues to which they are relevant. We close this work with a summary of our conclusions in Section VII.

II. An Overview of Random Peer-to-Peer Communication

In MRS research, *what* is communicated by a robot and how that communicated information affects the performance of a system usually is what is of interest. *How* that information is communicated often is somewhat of an afterthought; as

1-4244-0602-1/07/$20.00 ©2007 IEEE.

long as the information from A to B in a timely fashion and uncorrupted without degrading system performance, the medium of choice is of little concern.

Conversely, our research into RP2P communication is aimed at identifying efficient and robust means of moving information about a decentralized MRS. We are concerned not with *what* actually is transmitted but with how it gets from A to B.

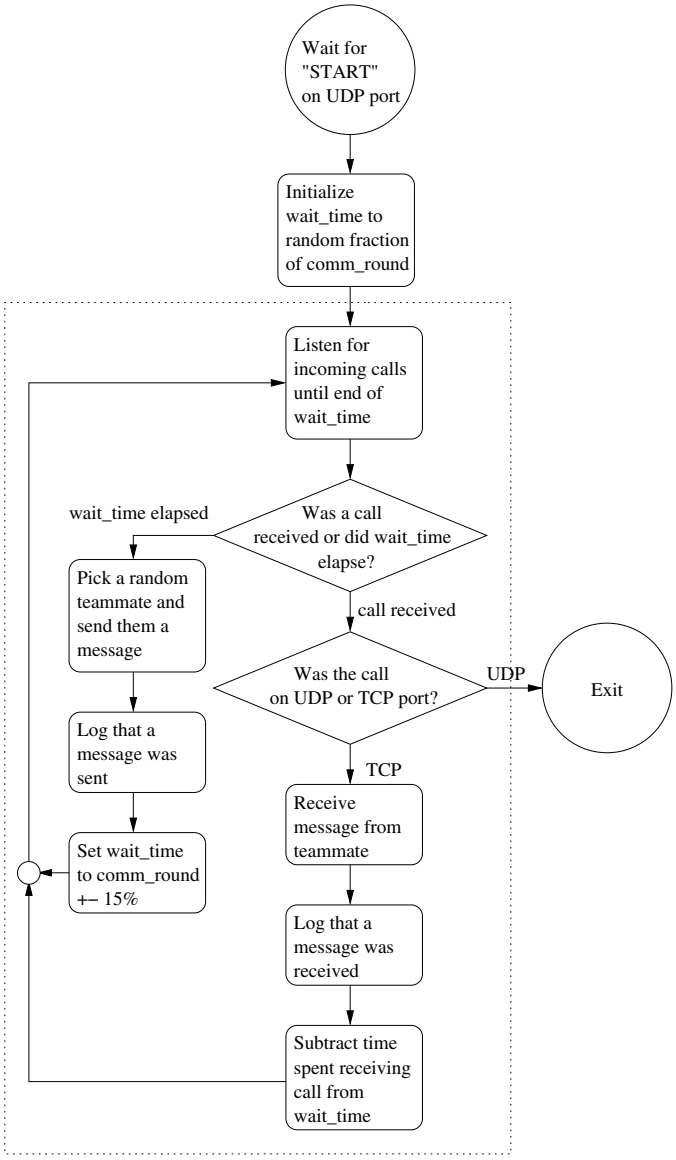

Fig. 1. This figure illustrates the behavior of the program that was used to control the individual robots that were used in this paper. The area enclosed in the dotted line is the core loop that carries out RP2P communication. It listens for messages from teammates for a period of time known as the communication round. At the end of every communication round, the program sends a message to a randomly chosen teammate. All sent/received calls are logged along with the time that they were sent/received.

Robots communicating via RP2P periodically send messages to randomly chosen teammates. These messages are sent directly from originator to recipient in a single hop. The periodicity of RP2P is captured in the notion of "communication rounds": a period of time over which a robot can be expected to transmit a message. We denote the length of a communication round by τ_{com}. Every τ_{com} seconds, each robot randomly selects a teammate and sends it a message. This all is carried out asynchronously; Robot-A's current communication round might end right now, while Robot-B might not send a message until some time in the future. All robots use the same value of τ_{com}. Thus, in a properly functioning RP2P system, messages will be sent continuously and yet all robots will have equal access to the wireless medium without any explicit centralized coordination. Unlike broadcast communication, RP2P messages are addressed to specific individuals at the IP-layer, so message filtering can be carried out by network interfaces without burdoning the robots' application layers.

Our earlier analysis of RP2P presented two interesting results [4]. First, information can be shared across a decentralized MRS in logarithmic time with respect to system population. Second, except for MRS where the population size is trivially small, the individual members of a MRS carrying out RP2P will experience the same volume of message traffic regardless of their system's population size. Specifically, the probability of a particular robot receiving k messages per communication round is relatively constant and rapidly converges to $\frac{1}{k!e}$ as the population size increases.

In our earlier analysis, it was assumed that an infinite bandwidth communication channel was available to the robots. Of course, in the real-world, this would not be the case and it is obvious that the value of τ_{com}, the system population-size and the bandwidth of the communication channel are related and likely governed by a relationship similar to Equation 1, where s_{msg} is the maximum message size, s_{wifi} is the wireless protocol's overhead that gets transmitted along with a message, n_r is the number of robots in a system, and B the bandwidth of the wireless channel.

$$(s_{msg} + s_{wifi}) \cdot n_r / \tau_{com} \leq B \tag{1}$$

Whether or not a physical MRS can be made to communicate via RP2P (that is, can the robots share a wireless channel simply by asynchronously exchanging messages with each other periodically as set by τ_{com}), and whether or not individual robots will be overloaded with messages are what we will be verifying in this work. The sharing of information in logarithmic time will implicitly be verified through the verification of these other two performance metrics.

III. Experimental Setup

In this section, we describe the hardware and software that were used to implement a physical system to analyze the behavior of real-world RP2P communication.

Our robots' controllers were based around the gumstix [sic.] embedded computer [5]. More specifically, we combined a "gumstix connex 200" embedded computer along with a "CF-stix" CompactFlash adapter and a Belkin F5D6060 CompactFlash 802.11B wireless card for each robot's controller. The connex 200 features a 200MHz Intel XScale processor, 4MB

of filespace and the Linux operating system. Our experiments were conducted on a private ad-hoc (peer-to-peer) wireless LAN. All of the robots remained stationary within a few meters of each other for the duration of a trial.

The robots were programmed in C using `gcc`. All communication was implemented with system calls to the standard `socket.h` library. The layout of the program is illustrated by the flowchart in Figure 1. The robots first carry out an initialization, which sets up sockets to listen on, etc., and then wait until they receive a "start" command broadcast over a UDP port from a central controller. Once the "start" command is received, the robots record their local time and treat this recorded time as t_o for the remainder of the trial. All peer-to-peer communication is carried out via TCP, which guarantees that all messages will arrive at their destination without error.

The robots are provided with a time to use as τ_{com} for the trial at run-time. Every $\tau_{com} \pm 15\%$ seconds, a robot will randomly select a teammate and send it a message. The $\pm 15\%$ variation in the periodicity of transmissions keeps the individual robots' transmissions asynchronous with respect to their teammates' transmissions. In between their transmissions, the robots receive any messages sent to them by their teammates. For every message that a robot sends or receives, it logs the time at which it was sent/received along with the IP of the sender/recipient and a rolling 8-bit tag. The tag is used to identify individual messages in both their originator's sent-log and their recipient's received-log. The time logged for a sent message is the time that the system call to `send()` returns. The time logged for received messages is the time at which the system call to `recv()` is made, which is made immediately following the return from the call to `accept()`. A trial ends when a "stop" command is received over the UDP port *after* the initial "start" has been received. Aside from the "start" and "stop" commands, there is no UDP traffic during a trial and the central controller remains silent; its sole purpose is to begin and end experimental trials.

In particular, we are interested in effects of three experimental variables on the performance of RP2P. These are: the length of a communication round (τ_{com}), the number of robots that compose a system and the lengths of the messages that are exchanged by the robots. We ran all 42 combinations of the seven values for τ_{com}, three population sizes and two message sizes listed in Table I. Each experimental trial was run sufficiently long to allow each robot to send at least 100 messages; trials often were allowed to run longer.

τ_{com}	Population Size	Message Length
1.0s	2 robots	4 bytes
0.5s	6 robots	256 bytes
0.2s	10 robots	
0.1s		
0.05s		
0.02s		
0.01s		

TABLE I
EXPERIMENTAL VARIABLE VALUES

IV. METHOD OF ANALYSIS

The first step in analyzing the data from a given trial was to merge the robots' individual trial logs into a single master log. It was assumed that any lack of synchronization between the robots' individual trial clocks was negligible.

From the merged logfiles, the mean time between transmissions by individual robots was computed, along with its variance. We will refer to the measured time between a robot's transmissions as τ'_{com} to differentiate it from τ_{com}, the time that the robots were instructed use as their inter-transmission time.

We next compute the probabilities of a robot receiving different numbers of messages per communication round. In our earlier work, to simplify our theoretical models, we assumed that all of the robots' communication rounds were synchronized. However, one of the advantages of RP2P is that it is *asynchronous*. We define a function $f_{\tau'_{com}}(t)$ for a given trial as shown in Equation 2. Note that τ'_{com} is used in Equation 2 rather than τ_{com}.

$$f_{\tau'_{com}}(t) = \begin{cases} 0 & t < 0 \\ 1 & 0 \leq t \leq \tau'_{com} \\ 0 & t > \tau'_{com} \end{cases} \tag{2}$$

Next, we define a second function, $f_{rec}(t)$, that is the sum of a set of Dirac delta functions; one centered about each time in the log that a message was received by the particular robot under analysis. The definition of $f_{rec}(t)$ is given by Equation 3.

$$\begin{array}{l} \forall a \text{ such that a msg was recv'd at time } t = a, \\ f_{rec}(t) = \sum_{i=a} \delta(t - i) \end{array} \tag{3}$$

Recall that the integral of the Dirac delta function is unity. We define a final function, $f_{msgs/\tau'_{com}}(t)$ that is the convolution of Equations 2 and 3, given in Equation 4. $f_{msgs/\tau'_{com}}(t)$ represents the number of messages received by a robot during the communication round that begins at time t. It has a stair-step shape, the amplitude of which indicates the rate at which messages are being received.

$$f_{msgs/\tau'_{com}}(t) = f_{rec}(t) * f_{\tau'_{com}}(t) \tag{4}$$

An example of $f_{msgs/\tau'_{com}}(t)$ along with the received messages that produced its shape is given in Figure 2. We calculate the probabilities of receiving various numbers of messages per communication round by computing the normalized histogram of $f_{msgs/\tau'_{com}}(t)$ for $0 \leq t \leq t_f$, where t_f is the time of the end of the trial.

V. EXPERIMENTAL RESULTS

In this section, we present the results of our experiments. In the next section we put these results into context with the broader issues that surround RP2P communication.

In all of our experiments, we found that there was no difference between the systems that were exchanging 4-byte messages and those that were exchanging 256-byte messages.

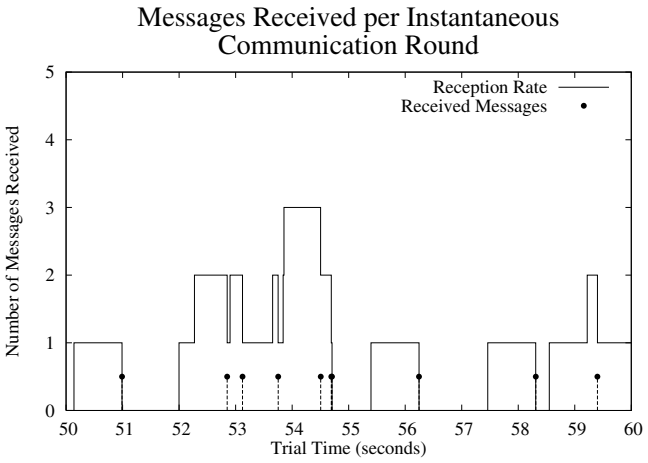

Fig. 2. This figure plots the number of messages received per communication round by a robot as well as the times at which the messages were received. The stair-step function is found by treating the impulses as Dirac delta functions and computing the convolution of these delta functions with a rectangular function of unity height and length equal to the duration of a communication round. The height of the convolution is equal to the number of messages received in the communication round starting at that time.

All of the figures presented in this paper correspond to the 256-byte trials. Figure 3 compares $\overline{\tau'_{com}}$ for the various trials that were conducted. Ideally $\overline{\tau'_{com}}$ should be equal to τ_{com}. For each system population, there seems to exist a minimum $\overline{\tau'_{com}}$, regardless of the value of τ_{com}. Second, the minimum $\overline{\tau'_{com}}$ appears to be a function of system population size. The variance of $\overline{\tau'_{com}}$ ($\sigma^2_{\tau'_{com}}$) increases suddenly when $\tau_{com} < \overline{\tau'_{com_{min}}}$.

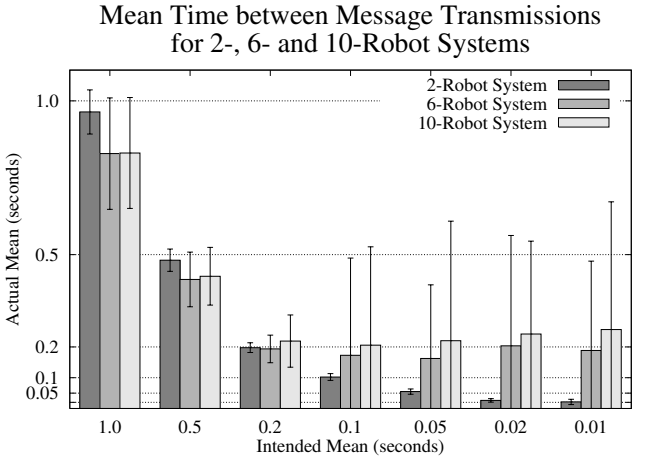

Fig. 3. This figure plots the average time between an individual robot's message transmissions. The horizontal axis indicates the intended time between message transmissions (τ_{com}) and the vertical axis indicates the actual mean time between transmissions. Note that, regardless of the intended value of a communication round, there is a minimum observed communication round for all three system populations, and that this minimum is population dependent.

For any given communication round, the actual time that a robot would wait before transmitting a message could differ from τ_{com} by as much as $\pm 15\%$, so in the ideal case, we cannot expect $\sigma_{\tau'_{com}}$ to be zero. If we treat the length of a communication round as a random variable with uniform distribution over $[\tau_{com} - 15\%, \tau_{com} + 15\%]$, then we know that $\sigma^2_{\tau'_{com}} = (1.15\tau_{com} - 0.85\tau_{com})^2/12$. Figure 4 plots $\sigma_{\tau'_{com}}/\overline{\tau'_{com}}$ along with the expected value of the normalized standard deviations. It clearly can be seen that the curves corresponding to the three system populations diverge from the expected value of $\sigma_{\tau'_{com}}/\overline{\tau'_{com}}$ at the largest value of τ_{com}. The lines corresponding to the 6- and 10-robot systems track the predicted standard deviation well until the communication round drops below 0.2 seconds, at which point they rise rapidly. The same phenomenon occurs in the 2-robot system when τ_{com} is set below 0.02 seconds. Note that the curve corresponding to the 10-robot trial rises slightly before that of the 6-robot trial. Were more trials conducted with τ_{com} varied about the range $[0.2, 0.5]$, we suspect that we would see the 10-robot curve begin to rise steeply slightly earlier than it does in Figure 4. The sudden increase in standard deviation indicates that messages no longer are being transmitted regularly and thus the assumptions required by our earlier theoretical analysis of RP2P no longer are satisfied.

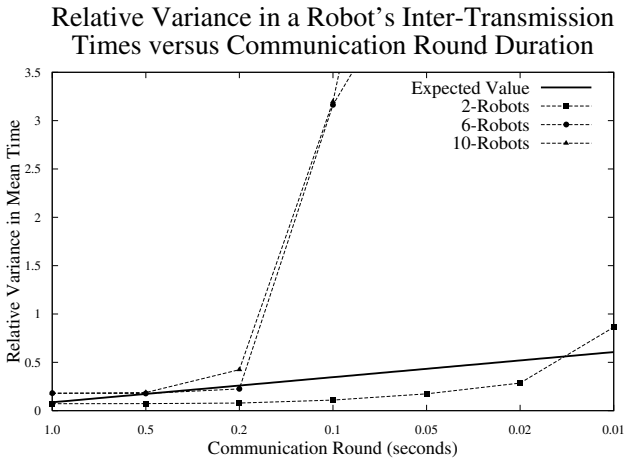

Fig. 4. This figure plots the standard deviation of the mean time between a robot's transmissions divided by the mean time between transmissions ($\sigma_{\tau'_{com}}/\overline{\tau'_{com}}$) versus the mean time between transmissions ($\sigma_{\tau_{com}}$). The divergence of the curves corresponding to the experimental trials from their expected value indicates that the periodic message passing of RP2P has broken down. In these cases, the system is not performing as it should.

Using Equations 2-4, we compute the probabilities of individual robots receiving 0-3 messages per communication round. The theoretical analysis of RP2P communication developed in our earlier work predicts that the probabilities of receiving 0-3 messages per communication round should be those plotted in Figure 5. We also include the asymptotes to which all of the plotted message-reception probabilities should converge in all of these plots.

Figure 6 plots the probabilities of receiving 0-3 messages per communication round as measured from the trials in which τ_{com} was 1.0 seconds. The figure's similarity to Figure 5 is clear. This resemblance, along with the knowledge that the individual robots in the system were behaving as we

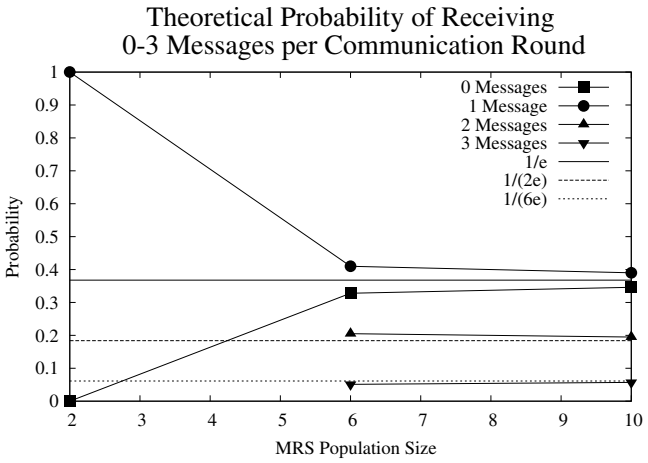

Fig. 5. This graph plots the theoretical probabilities of receiving 0-3 messages per communication round assuming that an infinite bandwidth channel is available over which to send the messages. Our earlier work demonstrated that the probability of receiving k messages per communication round quickly approaches $\frac{1}{k!e}$ as system population size increases. If the physical systems described in this paper are functioning correctly, the data from their message logs should produce graphs that are very similar in appearance to this one.

assumed confirms that RP2P communication is performing as our theory predicted in all of these trials.

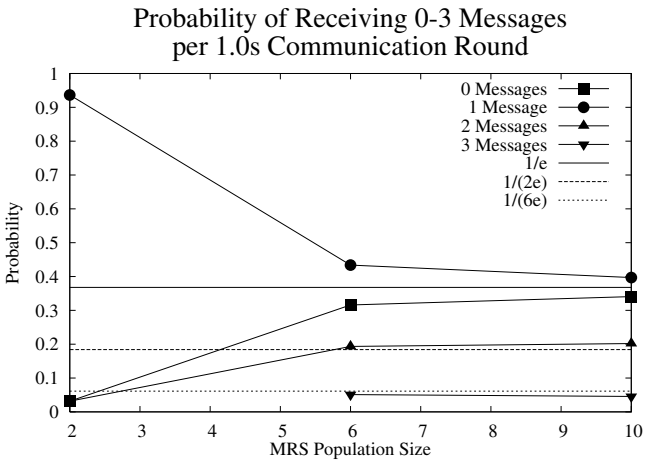

Fig. 6. This figure plots the measured probabilities of an individual robot receiving 0-3 messages per communication round when the duration of a communication round was set to 1.0 second during RP2P communication. The system is performing as it should, as indicated by this figure's resemblance to Figure 5.

We next present the measured probabilities of receiving 0-3 messages per communication round when the communication round was set to 0.1 seconds. These probabilities are plotted in Figure 7. The basic shape of the graph is similar to the theoretical predictions given by Figure 5 except that, for the 6- and 10-robot systems, the probability of receiving zero messages per round is elevated while the probabilities of receiving non-zero numbers of messages are depressed. Recall that for values of τ_{com} less than 0.2 seconds, the robots in the 6- and 10- robots did not regularly transmit messages as required by RP2P, nor were they able to send their messages as often as they were instructed to.

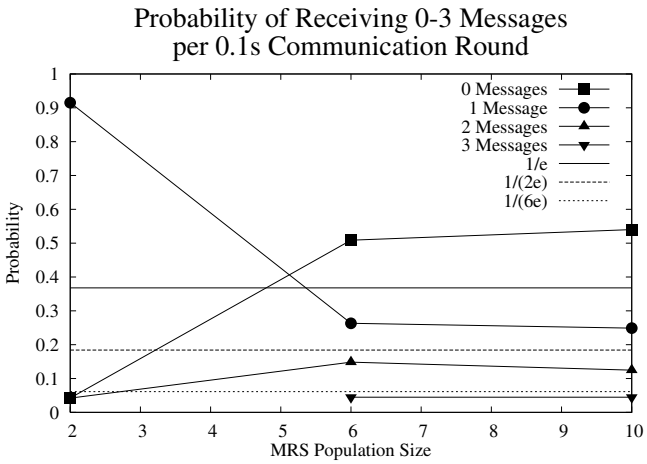

Fig. 7. This figure plots the probabilities of individual robots receiving 0-3 messages per communication round when τ_{com} was set to 0.1 seconds. The probabilities corresponding to the 6- and 10- robot systems are distorted. In those systems, the probability of receiving no messages is elevated while the probabilities of receiving any messages are diminished.

Finally, we present a plot of the probabilities of receiving 0-3 messages per communication round when τ_{com} was set to 0.01 seconds. From Figure 4, we can see that, for all three system populations, none of the robots were able to send their messages as frequently or as regularly as they were instructed to. Correspondingly, we can see that the probability of receiving zero messages is inflated above the theoretical prediction for all populations.

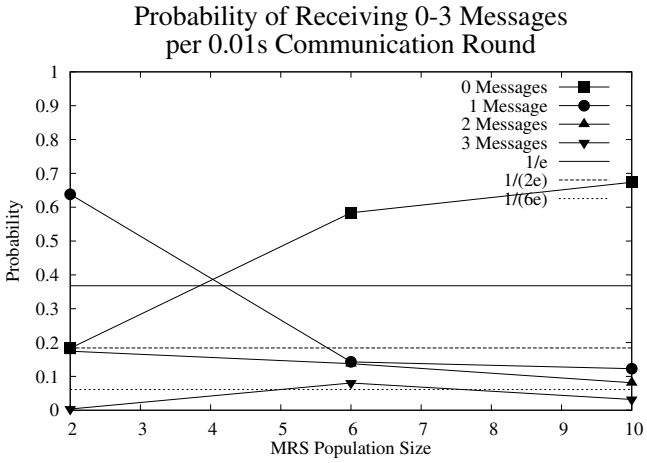

Fig. 8. In this final graph, we plot the probabilities of receiving 0-3 messages when τ_{com} was set to 0.01 seconds. In all of the trials with a 0.01 second communication round, all three system-populations behave abnormally; the probability of receiving no messages per communication round is inflated for all of them.

VI. Discussion

In this section, we will discuss the significance of the experimental results presented in the preceding section. Table

II summarizes the performance for each of the trials that we conducted with respect to the success or failure of the system to exhibit proper RP2P behavior. We will divide our discussion into two parts. First, we will discuss trials in which RP2P communication performed as expected. Then we will cover those situations in which the system's behavior was not as intended. We labeled a trial a success if $\sigma_{\tau'_{com}}/\overline{\tau'_{com}}$ is close to its predicted value (refer to Figure 4) and if $\tau_{com} \approx \tau'_{com}$. No differences were found between the 4- and 256-byte trials with regards to the success or failure of a configuration.

τ_{com}	2-Robots	6-Robots	10-Robots
1.0s	success	success	success
0.5s	success	success	success
0.2s	success	success	success
0.1s	success	failure	failure
0.05s	success	failure	failure
0.02s	success	failure	failure
0.01s	failure	failure	failure

TABLE II

Our experimental results demonstrate that RP2P is a viable communication scheme for MRS as long as the value of τ_{com} is made large enough to accommodate system population size and wireless channel bandwidth. In the cases where we labeled a trial a success, the measured probabilities of robots receiving different numbers of messages per communication round were precisely what our theoretical analysis of RP2P predicted. That the data 4-byte and 256-byte trials were virtually indistinguishable suggests that the maximum usable message size is significantly larger than 256 bytes (at least before the message size has any measurable effect on system performance). The results of Anastasi et al. suggest that we should see no degradation in performance if we increase the sizes of the individual messages sent by the robots up to 1024 bytes or more [6]. Given their detailed analysis of 802.11B ad hoc networks, message sizes as large as 10 kilobytes would not be unreasonable with our current system under ideal conditions. Note that more advanced wireless networking protocols likely would increase this upper bound significantly.

In those trials where RP2P performance degraded, we believe that it was the overhead of 801.11B that primarily was responsible. It must be pointed out, though, that RP2P does not fail catastrophically when τ_{com} is set too low. The wireless channel simply slows down as the robots try to transmit messages too frequently. Since the standard system call `send()` blocks (e.g., does not return until the message has been buffered by the kernel), the system simply slows down to accommodate the weakest link in the chain. The main drawback of trying to run RP2P too quickly is the system will lose the elegant, predictable behavior that otherwise would have existed.

Since there are no performance gains to be had by setting τ_{com} too low (e.g. trying to communicate too frequently), it would be desirable to identify the minimum τ_{com} that a system could sustain. Figure 4 illustrates that setting τ_{com} too low could be detected easily on-line via the variance in inter-transmission time. The members of a decentralized MRS communicating via RP2P should be able to adapt their working value for τ_{com} on-line to maximize the data transfer over the wireless medium.

VII. Conclusions

In this paper, we have demonstrated via physical experiments with real wireless networks that Random Peer-to-Peer communication is practical with commonly available, off-the-shelf components. We successfully implemented RP2P with 2-, 6- and 10-robot systems and our results suggest that considerably larger MRS could utilize this communication scheme. We identified two key variables that affect the performance of RP2P: the number of robots that populate a system and the rate at which they transmit their messages. Additionally, a third variable, message size, should play a part in the performance of the scheme, but the values investigated in our experiments produced inconclusive results.

We also found that MRS of all population sizes can degrade RP2P performance by attempting to transmit messages faster than the chosen wireless protocol can handle. Robots within a MRS are able to detect whether or not the rate at which they are sending messages is violating the assumptions of RP2P, so there is no reason why adaptive RP2P could not be implemented that would adjust in real-time to maximize data transfer rates while retaining the elegant, predictable behavior of RP2P.

802.11B is by no means the ideal implementation of ad-hoc networking for RP2P. It is, however, inexpensive, readily available and widely supported. There also is no reason why any other wireless protocol that supports peer-to-peer networking could not also be used to implement RP2P. The results detailed within this paper suggest that most existing MRS could utilize RP2P to enable robust robot-robot communication, and therefore cooperation, without any hardware modifications.

References

[1] G. Dudek, M. Jenkin, and E. Milios, *Robot Teams*. A K Peters, Ltd., 2002, ch. A Taxonomy of Multiple Robot Systems, pp. 3–22.

[2] O. Holland and C. Melhuish, "Stigmergy, self-organisation, and sorting in collective robotics," *Journal of Adaptive Behaviour*, vol. 5, no. 2, pp. 173–202, 1999.

[3] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, pp. 1–23, 1997.

[4] C. A. C. Parker and H. Zhang, "An analysis of random peer-to-peer communication for system-level coordination in decentralized multiple-robot systems," in *Proceedings of the 2006 IEEE.RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*, 2006.

[5] "Gumstix inc. website," http://www.gumstix.com, September 2006.

[6] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "Ieee 802.11b ad hoc networks: Performance measurements," *Cluster Computing*, vol. 8, no. 145, pp. 135–145, 2005.