# Evaluating Control Strategies for Wireless-Networked Robots Using an Integrated Robot and Network Simulation

Wei Ye[1,2] Richard T. Vaughan[1] Gaurav S. Sukhatme[1] John Heidemann[1,2]
Deborah Estrin[2*] Maja J. Matarić[1]

weiye,vaughan,gaurav,johnh,estrin,mataric@usc.edu
[1]Department of Computer Science
[2]Information Sciences Institute
University of Southern California
Los Angeles, CA 90089-0781

## Abstract

*Wireless communication is an enabling factor in multiple mobile robot systems. There is significant interaction between robot controllers and communications subsystems. We present a method for evaluating combined robot control/communication strategies for a team of wireless-networked robots performing a resource transportation task. Two alternative controller designs are compared under established communication and radio propagation models. For each we measure the overall performance of the robot team including the cost of communication. The study illustrates how our evaluation tools can be used for designing controllers for robots operating in wireless communication environments.*

## 1 Introduction

Successful control and coordination of a group of wireless-networked robots relies on effective inter-robot communication. Conventional robot controllers should be robust with respect to uncertainty and variation in sensing and actuation; controllers that exploit information distributed over a wireless network must also contend with imperfect communication.

In such a system the network is part of the robots' environment and can contribute to the system's success or failure. At the most simple level, communication may succeed or fail between two nodes; controllers should be designed to show graceful degradation as the probability of connectivity between nodes decreases. Controllers have been designed with this in mind [1]. However, there are other issues that are more subtle but still significant.

With increased use of wireless communications, transmission power can be a significant system cost, particularly for very small or very long-lived robots. Wireless Ethernet is far more energy expensive than wired Ethernet, but is a natural medium for mobile robots. Minimizing transmission time saves power for useful work.

Robots typically operate under strict real-time constraints; fast navigation and dynamic environments require that control inputs are acquired in a timely manner. Heavy load on a wireless network increases the average data transmission time, reducing the frequency response of controllers. Thus it may be impossible or more difficult to exploit some kinds of information, reducing the capability of the controller. Reducing load by more efficient communication can decrease latency and allow robots to be more responsive to dynamic environments.

Bandwidth itself may be a precious resource if the robot's task involves transmitting data to a user *e.g.* live video. Controller data is unwelcome overhead on the shared communications channel. Efficiency becomes increasingly important when scaling to large numbers of nodes.

Communication therefore has implications for controller robustness, efficiency, and capability. To save power and reduce latencies, communication should be as efficient as possible. We believe that the best efficiency can be achieved by considering the interaction between controller and communication channel; controllers should be designed to take into account protocol characteristics and propagation conditions, and specialized protocols can be designed for mobile robot

*also at the Computer Science Dept., University of California, Los Angeles, CA 90095-1596 [destrin@cs.ucla.edu].

applications. To this end we have integrated our established network and multi-robot simulators. The resulting tool allows us to evaluate networked controller designs in a convenient real-time environment, with the aim of reducing the number of costly real-world experiments.

This paper describes our simulator and shows how it can be used to inform the design of controllers for a team of mobile robots. We evaluate two controllers that use slightly different high-level inter-robot communication strategies. The results show their performance is similar, but that one incurs significantly more communication cost than the other.

The main contribution of the paper is: (1) showing that a more detailed communication model does affect robot control algorithms; (2) a hybrid simulator can be used for identifying (narrowing) parameters for study.

# 2 Integrated Simulators

We have combined two existing special-purpose simulators with which we have extensive experience:

**Arena** [3] is a multi-robot simulation tool developed at the Robotics Research laboratory at the University of Southern California [`http://robotics.usc.edu`]. It used for teaching and research into control and coordination of multi-agent systems.

Arena simulates the movement and sensing of many ($0 < N < 254$) robots in a two-dimensional area. Robots co-exist in a environment of static obstacles, specified by a user-defined bitmap. Arena spawns an instance of Golem [4] for each simulated robot; Golem provides a powerful publish-and-subscribe interface to a robot's resources, accessed over the network via TCP sockets. Robot controllers are implemented as independent client processes, each obtaining its sensor data and writing its motor commands through a Golem. Golem and Arena communicate asynchronously through a shared memory segment. At each simulation step Arena moves its robots and checks for object collisions. Robots' sensor readings are recalculated at intervals that closely match the real devices.

Arena provides no mechanism for symbolic inter-controller communication. In past experiments we have used UDP [1] and TCP sockets for this purpose.

The **Network Simulator (*ns*)** [5, 6] is a widely used network simulation tool in the networking community. It has been developed for many years, and is now under the VINT project [7], a collaboration among USC/Information Science Institute, Xerox Palo Alto Research Center, Lawrence Berkeley National Laboratory, and University of California at Berkeley. *ns* implements a number of networking protocols for both wired and wireless communications. It is a powerful tool for studying detailed network behaviors at different communication layers.

In order to preserve compatibility and modularity we have built a minimal bridge between Arena and *ns* that allows us to run the two simulators side by side. Arena handles robot modeling and interaction with the physical environment while *ns* simulates robot communication protocols and radio propagation. Robot controllers drive virtual Arena robots and perform their communication through *ns*'s virtual network.

## 2.1 The Interface

Communication in *ns* occurs between spatially located **nodes**. Each node corresponds to a particular robot in Arena. We extended *ns* to connect to Arena's *Position-Server* via a TCP socket as shown in Figure 1. Arena periodically announces the location of each robot over this connection. When the *ns* object named 'Arena-IO' receives this information, it updates the position of each node accordingly.

*ns* was further modified to handle communication requests from external robot controller processes. *ns* creates a robot agent for each robot/node. Each robot agent sets up a TCP socket with its corresponding robot controller process. Once this connection is made, *ns* can send (receive) messages to (from) each external controller. Whenever a packet is sent from a robot controller, it is received by its corresponding robot agent in *ns* via this TCP connection. The robot agent then tries to send the packet to the robot agent on the destination node. *ns* determines whether the packet is lost due to *e.g.* collisions, a full interface queue, or bad propagation conditions. If the destination node successfully receives the packet, it immediately forwards the packet to its corresponding robot controller.

As we run all processes on the same machine, we assume that the various 'real' TCP connections are not significantly effecting the simulated communication dynamics. Because Arena simulates robots in real time, we assume that all components of the system are fast enough to keep up with real time. This restriction hasn't been a problem in our simulations of 6 robots on a modest 500MHz Pentium III computer. If ns falls behind real-time it will detect and report this fact.

Combining ns and Arena in this way is an example of simulator federation [8].

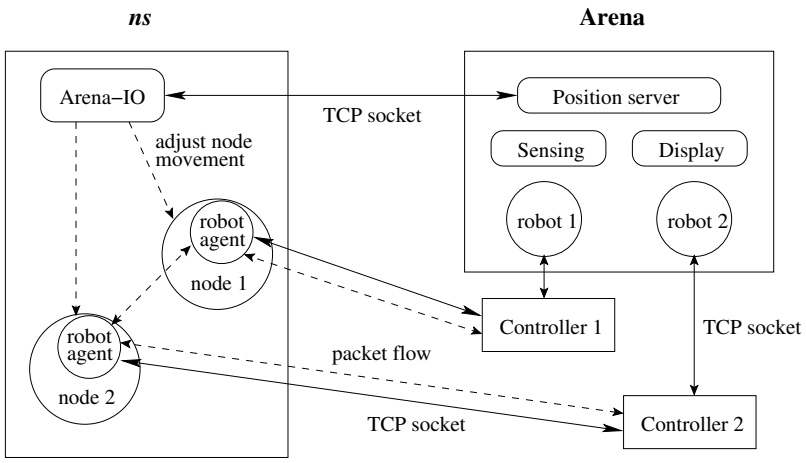


Figure 1: Integration of Arena and *ns*.

## 2.2 Radio Propagation Models

The ideal propagation model is readily available in *ns*. It basically represents the communication range as a circle around the transmitter. A typical ideal model is the Friis free space model [9], which predicts the received signal power at distance $d$ from the transmitter as

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{1}$$

where $P_t$ is the transmitted power. $G_t$ and $G_r$ are the antenna gains of the transmitter and the receiver respectively. $L (L \geq 1)$ is the system loss, and $\lambda$ is the wavelength. It is normal to select $G_t = G_r = 1$ and $L = 1$ in simulations.

An ideal model predicts the received power as a deterministic function of distance. In reality, the received power is a random variable due to multi-path propagation effects. A more general and widely-used model is called the shadowing model [9]. In order to evaluate robot controllers under different propagation conditions, we have implemented the shadowing model in *ns*.

The shadowing model consists of two parts; the first is the path loss model, which predicts the mean received power at distance $d$, denoted by $\overline{P_r(d)}$. It uses a close-in distance $d_0$ as a reference. $\overline{P_r(d)}$ is computed relative to $P_r(d_0)$ as follows.

$$\frac{P_r(d_0)}{\overline{P_r(d)}} = \left(\frac{d}{d_0}\right)^{\beta} \tag{2}$$

$\beta$ is called the path loss exponent, and is usually empirically determined by field measurement. From Eqn. (1) we know that $\beta = 2$ for free space propagation. For outdoor obstructed environments, typical values of $\beta$ lie between 3–5 [9]. Larger values of $\beta$ correspond to more obstructions and hence faster decrease in average received power as distance becomes larger. $P_r(d_0)$ can be computed from Eqn. (1).

The path loss is usually measured in dB. So from Eqn. (2) we have

$$\left[\frac{\overline{P_r(d)}}{P_r(d_0)}\right]_{dB} = -10\beta \log\left(\frac{d}{d_0}\right) \tag{3}$$

The second part of the shadowing model reflects the variation of the received power at a certain distance. It is a log-normal random variable, that is, it is of Gaussian distribution if measured in dB. The overall shadowing model is represented by

$$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta \log\left(\frac{d}{d_0}\right) + X_{dB} \tag{4}$$

where $X_{dB}$ is a Gaussian random variable with zero mean and standard deviation $\sigma_{dB}$. $\sigma_{dB}$ is called the shadowing deviation, and is also obtained by measurement. Its typical value in outdoor environment is 5–12 [9]. Eqn. (4) is also known as a log-normal shadowing model.

Previously Arena assumed perfect communications between robots. By forwarding communications through *ns*, we are now able to capture packet loss due to multiple factors, such as limited communication range, media contention, packet collision and overflow of the outgoing interface. The communication range is decided by the transmission power, propagation condition and the receiving threshold. The shadowing

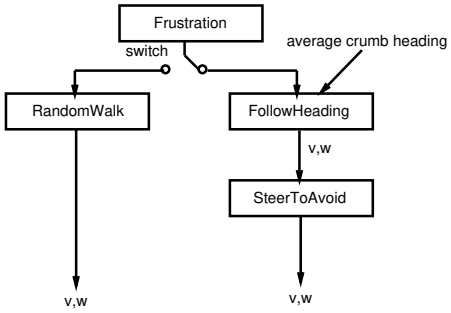


Figure 2: The 'Ant' robot controller

model extends the ideal circle model to a richer statistical model: robots can only communicate with reduced probability when near the edge of communication range.

# 3 Example Task

We consider an example application in which a team of mobile robots explores an initially unknown environment to locate a supply of resource. The robots must repeatedly collect units of resource and deliver them to their common start location. Our previous work has shown the utility of an ant-inspired trail following algorithm for this task, both in simulation [1] and real robots [2].

## 3.1 Robot Controller

Each robot records its movement as a list of landmarks in localization space. Each landmark records the robots' position, heading and the current time. Whenever a robot reaches the goal location and receives a unit of resource, it broadcasts its list of landmarks over the network so that other robots can follow. When a robot successfully returns home it also broadcasts its landmark list. Robots will follow the heading hint given by averaging the heading of any nearby landmarks. Common paths are created and reinforced as they are repeatedly followed. In this way, the robots tend to converge to a reasonably efficient path from source to goal and back.

The control strategy for the robots is composed from simple basis behaviors. These are shown in Figure 2. *FollowHeading* and *SteerToAvoid* work together to drive the robot towards the local average landmark heading while steering past most obstacles. If there are no nearby landmarks, the robot just drives forwards. An emergency stop mechanism switches off the wheel motors to prevent collisions. The simple steering strategy can be defeated by tricky arrangements of obstacles, so occasionally the robot will stop facing a wall or another robot. *Frustration* detects this condition by monitoring the forward speed; if the robot has been still for a fraction of a second, *RandomWalk* is activated. This behaviour will 'unstick' the robot from a tight situation by moving randomly into open space for a few seconds. Dead-ends (local minima) are a problem for this controller, in this work we only consider environments which do not have local minima. Our previous work [1] shows that if *Frustration* latches *RandomWalk* for long enough the robot successfully escapes most local minima. Note that the controller does not need to know the direction of the goal, but does need a localization estimate. The controller is described in more detail in [1].

## 3.2 Communication Strategies

We investigated a simple but important design choice: how to broadcast a robot's landmark list. One simple approach is to pack the entire sequence of landmarks into a long packet and send it only once. If a robot receives the packet it gets complete trail information. If the packet is lost due to bad propagation or other communication limits, some or all of the robots will not receive anything at all.

The extreme alternative is to send out the list as a sequence of short packets, each consisting of one landmark. The motivation behind this design is that even if some packets are lost others could still get through. So some robots may receive at least partial information about what trails to follow. A single landmark was chosen as the smallest self-contained meaningful packet.

We used the integrated Arena/*ns* simulator to evaluate these two controller design strategies. We hypothesized that the single-landmark broadcast strategy would permit more robust overall system performance, as partial trails would be transferred between robots even in poor transmission conditions. We expected this to be at the expense of greater network load, as there is some per-packet overhead. This is the strategy used in our previous work. We will see below that this decision was based on a naive understanding of the behaviour of the wireless network subsystems.

## 3.3 Method and Metrics

The simulator permits us to monitor and measure many aspects of the system, including some that are obscured in a 'real' implementation. For this study we

designed metrics to capture the 'success' of the robots at their task, and the 'communication cost' of the chosen strategy. We used these metrics to examine the anticipated trade off between communication usage and the robot's overall ability to complete the task.

We examined the performance of our control and communication strategies under two different network conditions. The first of these uses ideal propagation model to represent "good" propagation conditions. In this regime there is no packet loss due to propagation factors. However, it is still possible that some packets are lost due to collision or overflow on the interface queue. Second, we use the shadowing model to provide a more realistic "poor" environment in which packets may be lost due to bad propagation conditions.

When a robot agent in *ns* receives a packet from a robot controller, it allocates a UDP packet, fills in the received data and tries to send it to the destination node. The packet goes down the communication stack, i.e., IP layer, link layer, interface queue, MAC layer, physical layer and finally the wireless channel. Each layer may add its own header to the packet and thus increase the communication overhead. When the physical layer of the destination node receives the packet, it examines the signal power, which is determined by the propagation model. If it is below the receiving threshold, the packet is dropped.

The MAC layer we have used is the IEEE802.11 [12]. For each data packet, the MAC layer transmits several control packets (*e.g.* RTS/CTS/ACK) for the purpose of (packet) collision avoidance. These control packets are also counted as communication overhead. Furthermore, if these control packets are lost due collision or bad propagation, the MAC layer will try to retransmit them (or the data packet) again.

We use the following approach to examine the success of each controller. The resource transportation task simulation is repeated 40 times for each of the ideal and shadowing propagation models. The number of units of resource transported in each simulation is recorded as our success metric.

The communication cost is the ratio of bytes transmitted to the number of useful data bytes received:

$$\begin{aligned} \text{Cost} &= \frac{\text{Utilized bandwidth}}{\text{Effective data throughput}} \\ &= \frac{\text{Total bytes transmitted}}{\text{Total bytes of data received}} \end{aligned} \quad (5)$$

The total number of transmitted bytes is counted at the channel, which includes all control packets and data packets. The number of bytes of received data is counted by each robot agent.

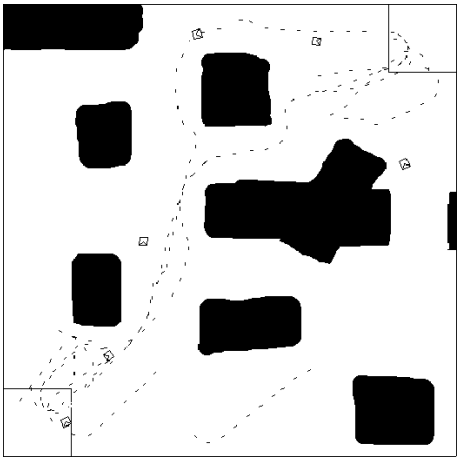

Figure 3: Screenshot of environment and working robots. The goal area is in the upper right corner and the home area is in the lower left corner. The dashes show the landmark trail generated by the robots.

# 4 Experiments

The experiments are carried out in a simulated square area of 33.33m on each side. Figure 3 is a screenshot of Arena during a representative trial. The home location is at the lower left corner, and the goal position is at the upper right corner. There are 6 robots in all and each trial runs for 3 minutes.

From Figure 3 we can see that the environment is quite obstructed, so the shadowing propagation model is appropriate. We choose the path loss exponent $\beta = 3$, and the shadowing deviation $\sigma_{dB} = 5$. The values of $\beta$ and $\sigma_{dB}$ are chosen from the ranges of their typical values [9]. They do not necessarily reflect the details our particular simulation environment or of a particular real environment.

By properly setting the receiving threshold in the physical layer, the shadowing model in our experiments allows 95% of packets to be correctly received at the distance of 20m from the receiver to the transmitter. Again, this does not necessarily reflect the details of a particular real environment but are generally representative of typical environments in which our robots are used. The values of $\beta$, $\sigma_{dB}$ and the receiving threshold can be adjusted to generate better or worse propagation environments.

The two robot controllers (using long packet and short packet transmission strategies) were tested with both the ideal propagation model and the shadowing model. In each case, the simulation runs for 30 complete trials. The dashed lines in Figure 3 represent the

| Propagation model | Robot Controller | |
|---|---|---|
| | Long Packet | Short Packet |
| Ideal | (36.77, 4.43) | (36.60, 4.17) |
| Shadowing | (36.57, 4.96) | (34.47, 4.27) |

Table 1: Units of resource transported (mean, standard deviation)

| Propagation model | Robot Controller | |
|---|---|---|
| | Long Packet | Short Packet |
| Ideal | (5.97, 0.15) | (29.29, 0.66) |
| Shadowing | (8.04, 0.66) | (41.68, 4.00) |

Table 2: Communication cost in bytes (mean, standard deviation)

landmark list currently held by one of the robots. A partial trail can be seen at the bottom middle of the picture; this is a result of the short-packet communication strategy in which it is possible for some landmarks in a trail to be received while the rest are lost.

### 4.1 Results

Table 1 shows the number of resource units transported by the group of robots. The data are expressed as (mean, standard deviation). Table 2 shows the communication cost in terms of the total number of bytes transmitted for a robot to actually receive one byte of data. Note that when the losses are high, such as in the shadowing model, the communication overhead is also high *e.g.* in the short packet/shadowing case shown in Table 2, approximately 42 bytes are sent for every 1 data byte actually received.

We use the data in Table 1 to evaluate the relative performance of the controllers. The t-test is performed on each pair of data sets: the null hypothesis ($H_0$) is that the short and long packet strategies are equivalent, producing results with the same distribution; the alternative hypothesis ($H_1$) is that they produce results with significantly different distributions. Table 3 shows the resulting $t$ values for resource and communication cost, indicating in each case if $H_1$ is favoured with a confidence level of 0.95 ($t > 1.67$).

The results of robustness evaluation are summarized as follows. Under ideal propagation conditions, the two controllers have similar performance. Is is not possible to discriminate between the short packet and long packet strategy. Under the shadowing propagation model, the controller using long packets is slightly better than the other. In this case, the statistical evidence suggests that long communication bursts are better for

| Controller | landmarks | $t$ | Difference |
|---|---|---|---|
| long packet | (18580, 1925.3) | 13.85 | Yes |
| short packet | (11939, 1719.5) | | |

Table 4: Number of landmarks (mean, standard deviation) received by robots under shadowing model

the transport task. The controller using long packets is robust under different propagation conditions. The controller using short packets has a slight performance degradation under poor propagation conditions. These results are evidence against our hypothesis that the short packet strategy would be more reliable.

Table 2 shows a significant difference in communication cost between the two cases. The controller using short packets has a communication cost approximately 5 times larger than that using the long packet. As discussed in Section 3.3, the reason is due to the substantial overhead for each packet. Another conclusion is that both controllers have increased communication cost in the poor propagation environment. However, the absolute cost of the controller using long packets is still quite small.

The above results show that the short-packet strategy performs significantly worse than the long-packet strategy under bad communication conditions. To find out the reason we examine the total number of landmarks received by robots. Table 4 compares the results of the two controllers under the shadowing model. From the table we see that the robots using the short-packet controller receives much less landmark data than those using the long-packet controller. From Table 2 we already know that the short-packet controller transmits far more packets, especially under bad propagation conditions. This greatly increases the possibility of collision and overflow of the interface queue. Therefore less useful data are transmitted overall.

## 5 Conclusions

This paper presents a method to evaluate control and coordination strategies for a group of wireless-networked robots. An integrated robot and network simulation tool was developed to allow investigation of the behavior of robot controllers combined with detailed communication models embedded in a virtual environment. In this study the simulation was used to study the performance of two robot control strategies under two alternative communication models. In the future we plan to study a variety of robot controllers to make broader and more accurate evaluation of robot

| Data set pairs | | Resources | | Cost | |
|---|---|---|---|---|---|
| | | $t$ | Difference | $t$ | Difference |
| long packet/ideal propagation | short packet/ideal propagation | 0.15 | No | 185.5 | Yes |
| long packet/shadowing propagation | short packet/shadowing propagation | 1.72 | Yes | 44.69 | Yes |
| long packet/ideal propagation | long packet/shadowing propagation | 0.16 | No | 16.47 | Yes |
| short packet/ideal propagation | short packet/shadowing propagation | 1.92 | Yes | 16.46 | Yes |

Table 3: t-test results for evaluating robustness and communication cost

controllers. The strategies studied here are extremal in that we sent either the minimum or maximum data in one packet. It would be interesting to examine the effect of varying the length of the message sent between these two extremes. In general, using long packets reduces the number of control packets. However, the cost of retransmitting an erroneous long packet is high. Another interesting work is to reduce this cost while keeping the low overhead of control packets. One way to do it is to internally fragment a long packet into smaller ones at the MAC layer [12].

The example application of resource transport demonstrates how to evaluate particular properties of a robot controller. The overall performance, robustness and communication cost are quantitatively studied and compared. The results showed that our assumption that a short-packet strategy would prove more robust was false. This provides evidence for our argument (Section 1) that there is a strong interaction between controller and network subsystem design. Our integrated simulation tool provides a unique means to investigate this interaction, with the goal of designing more effective control and communication strategies for mobile robots.

This study did not take advantage of a powerful feature we have built into Arena/*ns*. By inspecting Arena's bitmapped environment, *ns* can determine whether there is clear line-of-sight, *e.g.* no walls or other obstructions, between two nodes. Radio propagation conditions can be varied accordingly, to simulate the significant difference in signal strength and multipath effects between direct and indirect transmission. This feature adds significant richness to the simulation environment. Our future work will exploit this feature as we investigate robot controllers that consider propagation and network quality issues to guide their behaviour.

Wireless networks are becoming an increasingly common part of our environment. We believe that teams of autonomous robots can achieve new levels of competence and versatility by exploiting this technology. This will be best achieved by optimizing the interaction between a robot's controller and its communication subsystems. This will require innovations in both controller and protocol design. The Arena/*ns* simulator is a valuable tool for our work in this area.

## Acknowledgements

This work is supported in part by NSF grant ANI-9979457, DARPA grant DABT63-99-1-0015, and DARPA contract DAAE07-98-C-L028.

## References

[1] R.T. Vaughan, K. Støy, G.S. Sukhatme, and M.J. Matarić. Whistling in the dark: cooperative trail following in uncertain localization space. Proc. Int. Conf. Autonomous Agents, Spain, 2000. pp.187-194.

[2] R.T. Vaughan, K. Støy, G.S. Sukhatme, and M.J. Matarić. Blazing a trail: insect-inspired resource transportation by a robot team. Proc. Int. Symp. Distributed Autonomous Robot Systems, Knoxville, TN, USA. 2000. pp.111-120.

[3] http://robot.usc.edu/arena

[4] http://fnord.usc.edu/golem

[5] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. IEEE Computer, Vol.33, No. 5, 59-67, May, 2000

[6] http://www.isi.edu/nsnam

[7] http://www.isi.edu/vint

[8] R.M. Fujimoto, Time management in the high level architecture. Simulation, Vol. 71, No. 6, 388-400, December, 1998.

[9] T. S. Rappaport. Wireless communications, principles and practice. Prentice Hall, 1996.

[10] B.Holldopler and E. Wilson. The Ants. Springer-Verlag, 1990

[11] M. J. B. Krieger, J. Billeter, and L. Keller. Ant-like task allocation and recruitment in cooperative robots, Nature 406, 992-995, 2000

[12] LAN MAN Standards Committee of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specification, 1999. IEEE Std 802.11, 1999 Edition.