

Falke, Ryan R.

Multi-Agent Communication Framework

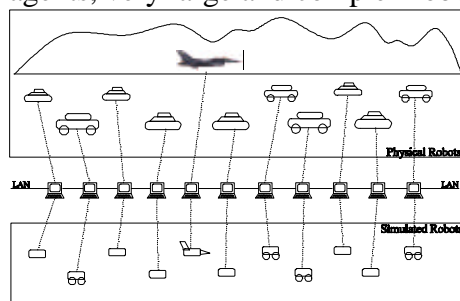
Faculty Mentor: James Archibald, Electrical and Computer Engineering

Multi-agent systems consisting of large numbers of autonomous robots are ideal candidates for tasks that could threaten human lives. However, the complexity and quantity of work required for precise coordination within such systems can be considerable. Our team developed a powerful but lightweight communication framework that significantly reduces the laborious efforts of coordination control. This program offers the ability to pass messages between robotic agents, and maintain knowledge of all robotic interactions. In addition, all the processes and communication nodes are location-independent – a complex task can be distributed amongst various agents. This paper briefly describes the framework designed and discusses some practical applications.

The Multi-agent Communication Framework, MCF, is designed on a hierarchical basis. This means there are different levels of command and control. There are two groups that help organize the tasks: daemons and agents. The daemons are the main points of communication, whereas agents are the workers that send the messages via the daemons. When a scenario begins at least one daemon must be running in order to facilitate communication between other agents. The daemon can act as a high ranking officer in the military; it can start and stop other daemons and agents as necessary.

In addition to the control structure we also need to have the ability to create the framework for both simulation and actual implementation. The ability to test ideas without having to construct costly hardware is vital in robotics research. MCF provides software tools that reduce development and debugging costs of multi-agent systems. After the software in a system has been tested in simulation, that same software can easily be ported to hardware. This is made possible through MCF by establishing a communications framework. The framework allows a programmer to write a program that the agent uses to “think.” This “thinking” code is basically the same on both hardware and software. The communications part of the program, which is different with hardware and software, is done by MCF. The programmer just adds a few lines of code to initialize the agent’s communications, and then uses a few library calls to send and receive messages. With this functionality the development time of applications is reduced.

With the ability to control hierarchical structures as well as communication between multiple agents, very large and complex robotic situations can be created and maintained easier. Consider



- Multi agent scenario

the challenges of developing a multi-agent system similar to that illustrated in Figure 1. In this scenario, an uninhabited air vehicle and robots are teamed together with a human operator (not shown) who monitors and controls various aspects of the team's behavior from a remote location. One of the challenges of this scenario is determining what information should be presented to the operator so that the efforts of the entire team can be directed appropriately. MCF provides the interface to the

operator so that he/she can track the progress of the entire team, switch to drive an individual robot, redirect the actions of a squad of robots working on a particular subtask, or issue a new high-level mission directive to the team.

MCF can also ease the burden of the intense computations necessary in multi-agent situations. Many of the different processes can be distributed amongst various computers. Distributed computing is advantageous by allowing computationally demanding code to be reassigned to a machine with more capacity, thus allowing the system to run more efficiently. Given this flexibility, computational resources can be scaled with the size of the multi-agent system.

While we accomplished most of our goals in designing MCF, the task is not yet complete. We realized a complete design, though we only implemented the skeleton. In winter semester of 2003 we constructed our own scenario similar to that of the one described above to test the new framework created. Our scenario involved a game of capture the flag between robots. There were two teams of five robots that had to traverse a maze and pick up the other team's flag. Only one human per team was involved in the game. Many of the robots operated autonomously, while others were controlled remotely by the human. In addition, the agents needed to be in constant communication with each other to work as a team.

This scenario proved to be a very good test bed for MCF. We were able to see how many agents the framework could maintain. At that time we supported up to sixteen agents simultaneously. MCF proved to be a significant aid in the design and implementation of this project. Many of the designers commented about the ease of use and how powerful the features were. The programmers were able to concentrate on the strategy of the game rather than how the agents were able to communicate.

In conclusion, MCF has the potential to have a significant impact on multi-agent engineering. The ability to focus on behavior programming rather than mundane communication problems is very valuable. However, the present project still has a long way to go. Other teams currently work to finish this original design. In the future the program will be more robust, offering many more features. Recently there have been some developments to make the original framework faster, and more efficient in message routing. The initial design of MCF has been very valuable and will pioneer other research in the robotics field.

I would like to thank the people who make the ORCA scholarship available to students who are interested in the pursuit of cutting edge technology. Because of the aid that they provided, we were able to create a solid product that will assist in the research and development of many other projects that follow. I would also like to thank Dr. James Archibald for his constant support and Matt Blank for his excellent ideas.