

## **A Two-level Distributed Interactive Simulation Architecture for Radar Frequency Assignment**

K. SRINIVASAN, J. CHERRY, N. D. CANNON, T. SCALF, S. RAMASWAMY

*Software Automation and Intelligence Laboratory*

*Department of Computer Science and Center for Manufacturing Research*

*Tennessee Technological University, Cookeville TN 38505*

**Abstract:** This paper presents a two-level architecture for the implementation of a distributed simulation environment for interference detection and frequency assignment in naval radar units. The Java-based simulation is designed as a two level architecture incorporating *i)* multiple radars on each ship controlled through a lower-level intra-ship interference control module and, *ii)* multiple ships in a group coordinated through a higher-level interference control module. Each ship's interference detection and control mechanism is composed of these two separate levels, called a control agent, dynamically coordinating with one another in eliminating interference problems. In addition, the control agent is a multi-threaded architecture that incorporates the maintenance of a distributed data base system, that contains periodically updated radar information.

*Submitted to:*

*The 1999 Summer Computer Simulation Conference*

*July 11-15, 1999*

# A Two-level Distributed Interactive Simulation Architecture for Radar Frequency Assignment<sup>\*</sup>

K. SRINIVASAN, J. CHERRY, N. D. CANNON, T. SCALF, S. RAMASWAMY<sup>1</sup>

Software Automation and Intelligence Laboratory, Department of Computer Science and Center for Manufacturing Research  
Tennessee Technological University, Cookeville TN 38505. Phone: (931)-372-3448 Email: srini@acm.org / srini@ieee.org

**Abstract:** This paper presents a two-level architecture for the implementation of a distributed simulation environment for interference detection and frequency assignment in naval radar units. The Java-based simulation is designed as a two level architecture incorporating *i*) multiple radars on each ship controlled through a lower-level intra-ship interference control module and, *ii*) multiple ships in a group coordinated through a higher-level interference control module. Each ship's interference detection and control mechanism is composed of these two separate levels, called a control agent, dynamically coordinating with one another in eliminating interference problems. In addition, the control agent is a multi-threaded architecture that incorporates the maintenance of a distributed data base system, that contains periodically updated radar information.

## I. Introduction

In last year's conference, a paper describing the benefits of a distributed simulation architecture for interference-free radar frequency assignments in Mobile Radar Units [1] was proposed. We also presented a simple prototype of the system to demonstrate the feasibility of an automated approach to oversee the interference detection and resolution process. It was shown that a distributed simulation environment was an excellent approach to study and prototype a system that allows the testing of various algorithms for the dynamic detection and resolution of interference in naval radar units. The simulation environment presented in that paper was based on the following assumptions: *(i)* each ship in the simulation was composed of one radar and always depended on a simulation server for its "knowledge" of other ships in the group. *(ii)* Each ship communicated with a centralized simulation server that executes the appropriate interference detection algorithm, which was then relayed back to the ships requesting interference detection. The resolution process was however, based on a locally autonomous approach, wherein each ship independently decides on the best alternative operating frequency based upon its local goals / needs without regard to the goals / requirements of the system.

In this paper, we present a two-level distributed interactive simulation architecture for the above problem. This approach has the following advantages to enhance the simulation environment: *i) Intra and Inter Ship Interference:* Interference resolution is *one* of the issues addressed by a control agent that has a two-level architecture, a lower level and a higher level. The lower level resolves the interference problem if the source of interference is within the ship. If the source of interference is not from within the ship, then the lower level passes the responsibility to the higher level. The higher level of the control agent detects the source (*outside*) of interference. If the source is a radar of another ship within the group, it may resolve the interference problem in one of the following modes, viz. Locally Autonomous, or the Negotiation based mode. *ii) Two-level Architecture:* A two level approach helps in balancing the local goals and the system goals and developing appropriate data structures and encapsulation techniques for addressing such concerns appropriately. Here the local goal of each control agent is to provide interference free operating frequencies to radar units on the ship and the system goal is to provide interference free operating frequencies for all radar units on all the ships within the group. *iii) Group Coordination Strategies:* Using the proposed architecture, it is easy to implement and study algorithms for group coordination. Different coordination and group management (arbitration, election of a group representative, determining group membership eligibility, rescinding / retaining group memberships, etc.) algorithms may be selectively implemented and comparatively evaluated. *iv) Localized Databases:* The two-level structure also incorporates a separate database associated with every control agent. By creating a separate thread of control for coordinating -i.e. report and maintain - database related information to legal group members, the simulation now provides for any individual control agent to assume control whenever it is decided that a group coordinator is no longer

---

<sup>\*</sup> This research is currently being conducted at the Software Automation and Intelligence Laboratory (SAIL) in the Department of Computer Science at Tennessee Technological University and is supported, in part, by grants from the National Science Foundation (#DMI 9896235) and the Center for Manufacturing Research at Tennessee Technological University.

<sup>1</sup> Please address all correspondences to Dr. S. Ramaswamy. Email: srini@acm.org / srini@ieee.org, Phone: (931) 372-3448.

available. v) *Generic Test-bed*: The two-level implementation will provide a generic test-bed for studying performance issues such as the available reaction time w.r.t. the speed of detection of an interference and its subsequent resolution, estimation of the percentage of false alarms and the consequences of subsequent actions, etc.

As stated earlier, each agent (ship) has one or more radars on it. Hence, interference for a given radar may be due to either of the following three reasons: *i*) another radar in the same ship, *ii*) radar of another ship in the same group, or *iii*) an external interference. In this paper the first two cases are referred to as intra-radar interference and inter-radar interference. The interference problem can be resolved by either changing the radar's frequency, bandwidth or position. It is however, desirable for the interference resolution process to change a radar's frequency.

The rest of this paper is organized as follows. Section II describes the Server agent architecture and its associated database.

## II. The Control Agent Architecture

Each ship's interference detection and control mechanism is composed of two separate levels, together called a control agent, which dynamically coordinates with other such control agents in eliminating interference problems. In addition, each control agent is designed to execute as a multi-threaded process incorporating a periodically updated database containing information about radars on the ship and within the group. The database is designed to provide complete information about all the radars on every ship in the group. As stated earlier, each control agent is a two-level architecture consisting of the lower and higher levels. Each level, in turn, has a different type of autonomy associated with it. The autonomy of the levels is broadly classified as: *i*) *Master-Slave mode*: In the master-slave mode, control is strictly hierarchically and top-down in nature using a table look-up technique. *ii*) *Locally Autonomous Mode*: In this mode, decisions are always made locally and based upon local goals of the agent without regard to the overall system goals. *iii*) *Negotiation based mode*: In this mode, the resolution of any potential conflicts is always through negotiation between the processes that experience the conflict, possibly with the aid of an arbitrator. At the lower level, the control agent resolves potential interference problems either through a master-slave or locally autonomous mode. At the higher-level, the control agent may adopt any of the three techniques. However, a master-slave configuration will almost always be the last choice while the negotiation mode would be the preferred course of action. Individual radars may, in turn, operate on any of the three modes.

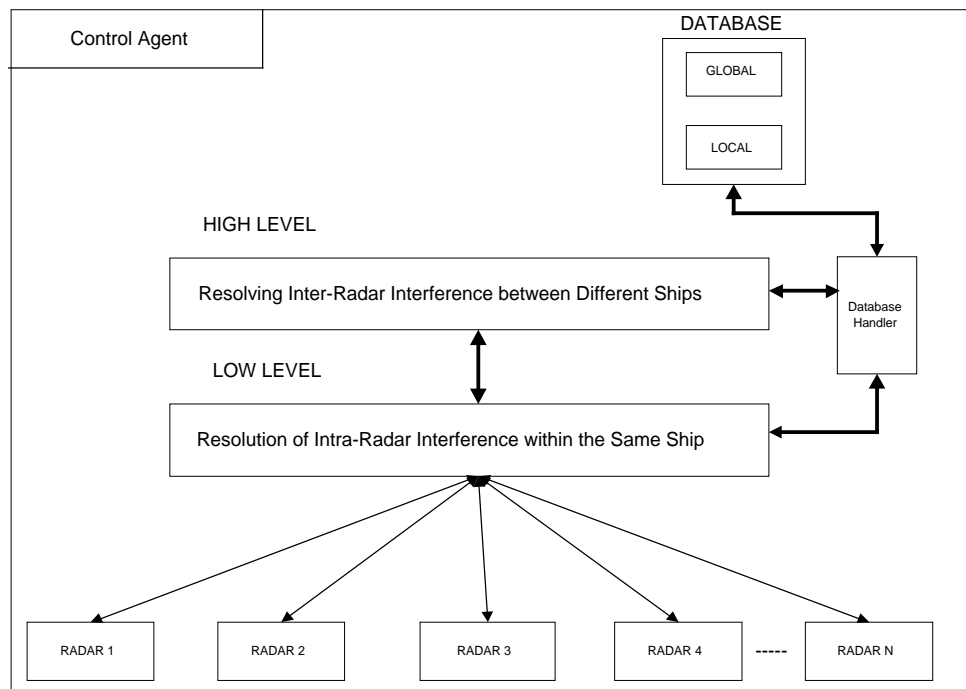
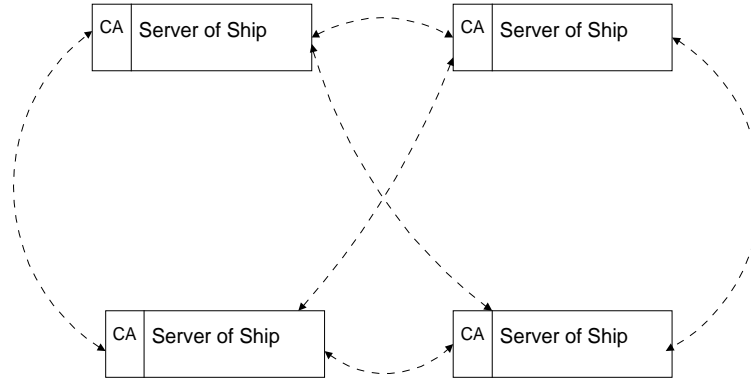


Figure 1. Two-level Control Agent Architecture



**Figure 2. Control Agent-based Communication Structure**

At the lower level, the control agent monitors and logs information about its radars continuously. If the server finds that a radar unit is experiencing interference, it informs the radar and determines if the interference is local, i.e., caused by some radar within the ship. In such a case, it assumes the responsibility of reassigning a frequency to either the interfering radar or both the radars involved if the radars are operating in either the master-slave or negotiation based modes. If the radars are operating in a negotiation-based mode, then the lower-level control agent serves the role of an information provider for the affected radars. As shown in Figure 1, the lower level control agent has complete access to the radar information about radars in the current ship from the database. Alternatively, if a radar unit within the ship does not cause the interference, then the control is transferred to the higher-level control agent. This is because the lower level control agent cannot communicate with other ships in the group and hence will not be able to modify the database with information pertaining to radar units on other ships. The higher-level module has complete access to the portion of the database that contains information about radars on other ships.

The higher level of the control agent can operate in any one of the following modes viz. Master-Slave, Locally Autonomous or the Negotiation based mode. In the Master-Slave mode the higher level cooperates and accepts decisions made by the group coordinator. Whenever it encounters an external interference it make a request to the group coordinator, that acts as a master control agent, for a new frequency. The master control agent may either allocate a new set of frequencies to all the radars on the ship that is experiencing interference, or assign a frequency to individual radar. In a locally autonomous mode, the higher level of a control agent in each ship does not communicate with other ships in the group to resolve its interference, but does signal to the rest of the group on any changes that it makes to its radar frequencies. Thus each ship may resolve its interference problem by randomly switching to a new frequency whenever it faces inter-ship interference problems. In a negotiation-based mode, each ship tries

```

public void run() {
    try {
        // Socket ready for connecting lower/higher level of control agent
        // Accept connection
        // Open Output Stream to send data to control agent
        // Open Input Stream to read data from control agent
        // String lower/higher level of control agent
        // String code from control agent for local or global access by higher/lower
        // level of control agent
        // Make sure everything is cleanly connected
        // Lower/Higher level of control agent reads data from local/global database respectively
        // Lower/Higher level may write into the local/global database respectively
        // Close connection
    }

    catch(IOException e) { }
}

```

**Figure 3. Control Flow within the Database Handler**

to identify the source of the interference. If the source is one of the members of its own group, it sends a message to the ship seeking a negotiation. When the ship causing the interference acknowledges the message, the two ships negotiate and settle for a frequency that is ideal for both of them. This system has a real-time co-ordination unlike the central frequency manager. The group communication structure of the control-agent-based implementation is shown in Figure 2.

The control agent implementation is conceptually split into three distinct processes. These include *i*) The database system, *ii*) The lower level control agent *iii*) the higher level control agent. In the remainder of this section, we will discuss each of these processes in detail.

### II. A. The Database

Each ship has a database (server) that runs in an infinite loop, which can connect to the lower or higher level of the control agent through a database handler. The database consists of two distinct information sources, *i*) information about the local radars and *ii*) information about the group's radars (i.e., radars on other ships). The local information in the database can be updated by lower level of the control agent.

The lower level control agent does not have the ability to modify the information about radar units on other ships within the group. This information can be accessed and updated only by the higher-level control agent. Whenever a radar unit information is updated on a ship, the handler broadcasts this information to the control agent on all ships. The control flow within the database is illustrated in Figure 3.

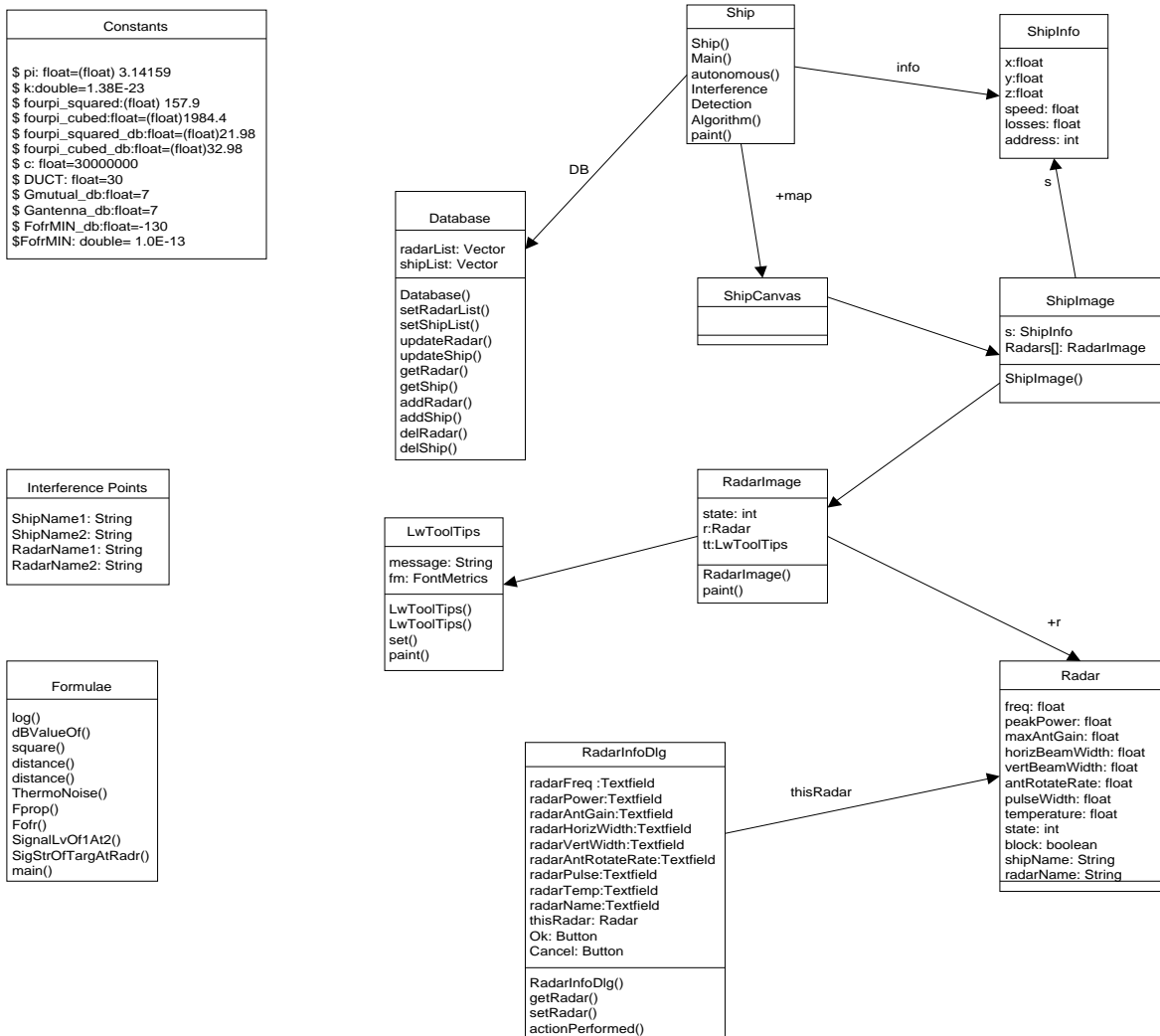


Figure 4. Lower-Level Control Agent Class Structure

## II. B. The lower-level Control Agent

The class hierarchy of the lower level control agent i.e. the Ship class, is shown in Figure 4. It uses the Database, ShipCanvas, ShipImage, RadarImage, LwToolTips, ShipInfo, Radar and the RadarInfoDlg classes. This class assumes the responsibility of checking whether a radar unit is experiencing local interference. If it detects that a radar unit is experiencing a local interference, then it informs the radar about the source causing it. Then the lower level of the control agent resolves this local interference appropriately. In such a case, it assumes the responsibility of reassigning a frequency to either the interfering radar or both the radar units involved if the radar units are operating in either the master-slave or negotiation based modes. If the radar units are operating in a negotiation-based mode, then the lower-level control agent serves the role of an information provider for the affected radar units.

The Database class is used to update and maintain the information about all the radar units on the ship. The options that are available in this class are shown clearly in the class diagram. Radar units can be added or removed from the ship and this class updates their operating parameters and position. The Ship class can read, write or update local information into the database only and does not have access to modify the global information within the database. The ShipCanvas class provides the GUI for viewing the radar and ship information, which are derived from the ShipInfo class. It uses the ShipImage, RadarImage and the LwToolTips classes. This class is used to display the ship with its radar units on it and its position. The LwToolTips class is used for selectively viewing additional information about the radars i.e. if the mouse pointer is pointed to a particular radar unit on the ship, the operating parameters and position of the corresponding radar are displayed. The RadarImage and the ShipImage classes are used to draw the radar units and the ships respectively. The ShipInfo class has information about the position and the speed of the ship. The Radar class has information about all the radar units and their operating parameters on the ship. It uses the RadarInfoDlg class for updating the operating parameters, position of a particular radar unit, or for introducing a new radar unit into the ship.

## II. C. The Higher-level Control Agent

The simulation is basically run as a client/server application. If a ship that serves as a coordinator is destroyed / loses communication with the rest of the group, then another ship is dynamically voted to be a coordinator. The current coordinator may be referred to as the active coordinator while all other ships can be thought of as passive coordinators. The higher-level control agent does frequency resolution involving radars on multiple ships. The control-agent runs in an infinite loop with a socket ready to accept a communication from any other group member. When a new control agent (ship) enters the group it sends an identifier to the coordinator for authorization. After authorization the client has to send all its parameters. The client information is registered by the coordinator and broadcast to the rest of the group only if its sends all its operating parameters to the coordinator after authorization. If two clients try to

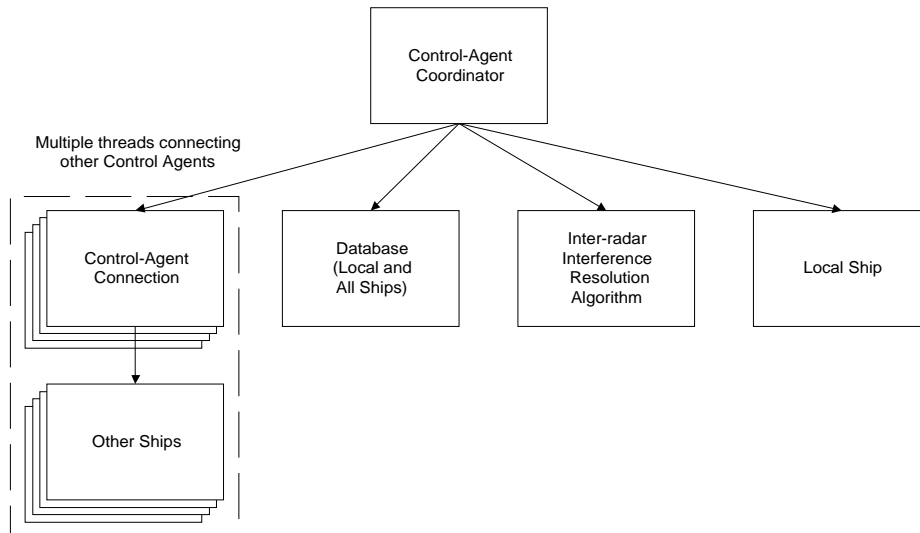


Figure 5. Higher Level Control Agent Class / Coordination Structure

connect at the same time, then the client that sends its complete message first gets registered, while the other client has to try and connect again, possibly with a different id if necessary.

The schematic of the higher-level control agent is shown in Figure 5. The control agent class is a multi-threaded implementation consisting of the control-agent connection, database, inter-radar interference resolution and the ship classes. The control-agent connection class initiates multiple socket connections to other members of the group for interference resolution. The database class has the complete list of all the radar units on the ship and about the radar units on all other ships. The higher-level control agent maintains and updates all database entries. The inter-radar interference resolution class handles interference problems referred to by the lower-level control agent. First using its current database information, it tries to identify the source ship and then initiates a communication with the control agent. The pseudo-code illustrating the three different ways in which the control agent can react when it experiences an inter-radar interference is shown in Figure 6.

```

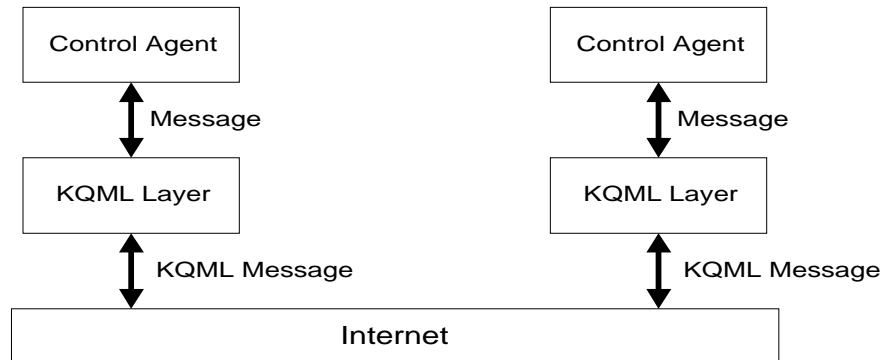
switch (mode) {
    case 1: { // Master-Slave mode
            // Connect to the group-coordinator and perform as directed by the coordinator
        }
    case 2: { // Locally Autonomous mode
            // Resolve the interference problem using the current information in database
        }
    case 3: { // Negotiation based mode
            // Identify the ship which hosts the radar in question, and initiate a negotiation with the
            // control agent of the ship
        }
}

```

**Figure 6. Interference Resolution at the Higher-level Control Agent**

### III. Implementation

The implementation is done in order to show the robustness of the two-level Control agent in assigning interference free frequencies to radar units in all ships in a distributed simulation environment. The KQML (Knowledge Query Manipulation Language) protocol developed by ARPA, which has been designed for supporting interaction and communication between intelligent software agents, is used for communication between higher level control agents. The implementation includes (i) Control Agent, (ii) Database (iii) Inter-radar Interference Resolution Algorithm and (iv) Ship classes. Each of the individual classes is discussed in detail in the following sections. Given below is a simple schematic representation of how control agents use KQML protocol for communication. If a control agent connects to another one, then it composes a message and sends it. This message goes to the KQML Layer, where the message content gets wrapped into the KQML message. The KQML protocol does not know anything about the message but for the beginning and end of the message. Thus the KQML message is sent to the other control agent. At the receiving end the KQML Layer parses the message and extracts the contents of the message and passes it to the control agent.



A simple KQML code in which a control agent starts a conversation with another control agent is given below:

Basically a KQML message consists of a performative and associated arguments that includes the actual message content. The performative is used to specify whether the message in the content field is a command, assertion or a query. Here the message content is "HELO" i.e. control agent A is starting conversation with control agent B. "ask-one" and "reply" are the performatives used by the Sender (S) and Receiver (R) respectively.

The meaning of the two performatives are given below:

*ask-one* : S wants an answer from R.  
*reply* : R is sending a reply for S's Query.

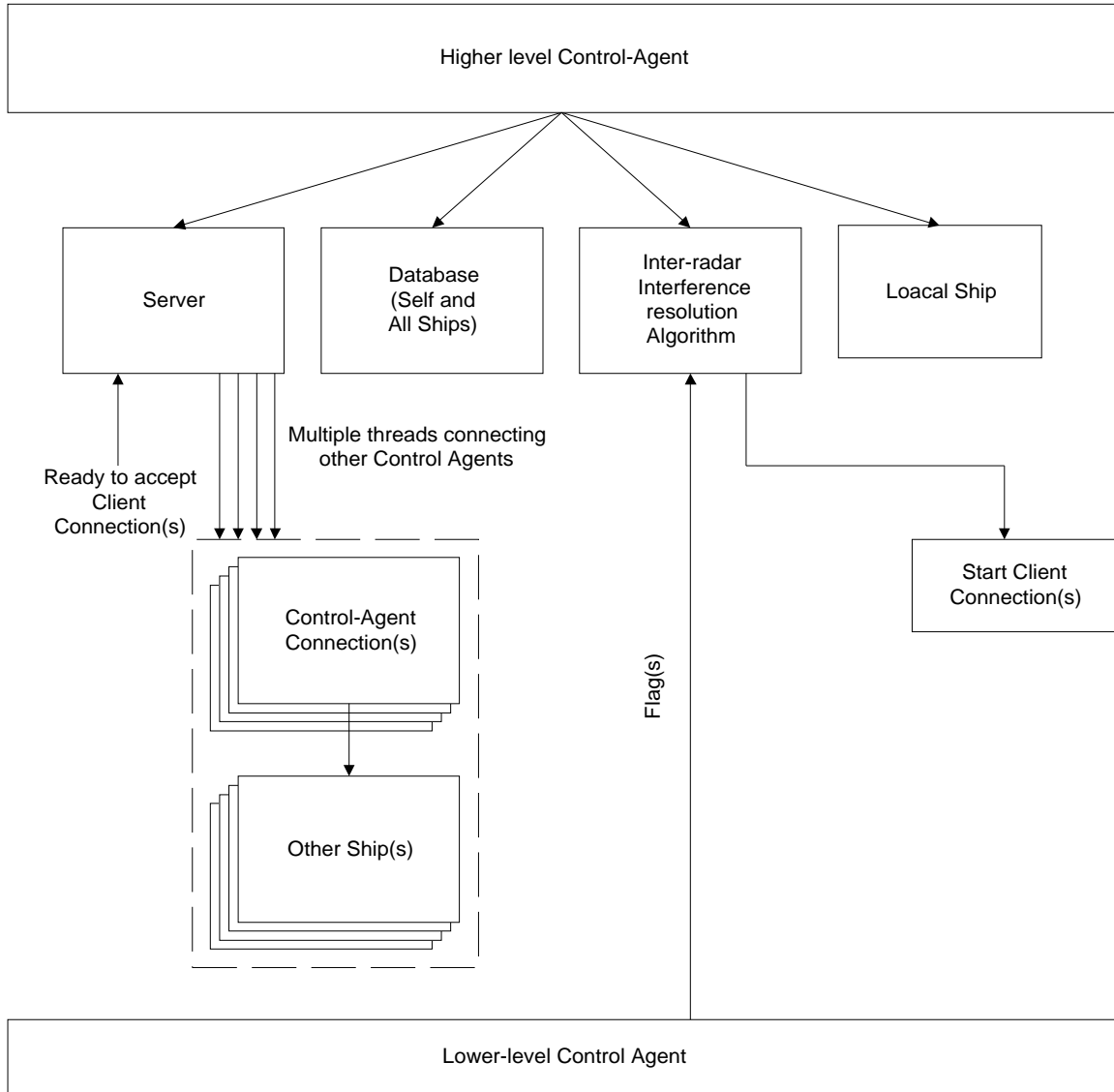
(ask-one  
: content ("HELO")  
: receiver control agent B  
: language KQML  
: ontology NEGO)

(reply  
: content ("HELO")  
: receiver control agent A  
: language KQML  
: ontology NEGO)

The other arguments receiver, language and ontology are used to specify as to which control agent the message is intended to, the language used and the context respectively. More information regarding KQML can be found at this site: <http://www.cs.umbc.edu/kqml/>.

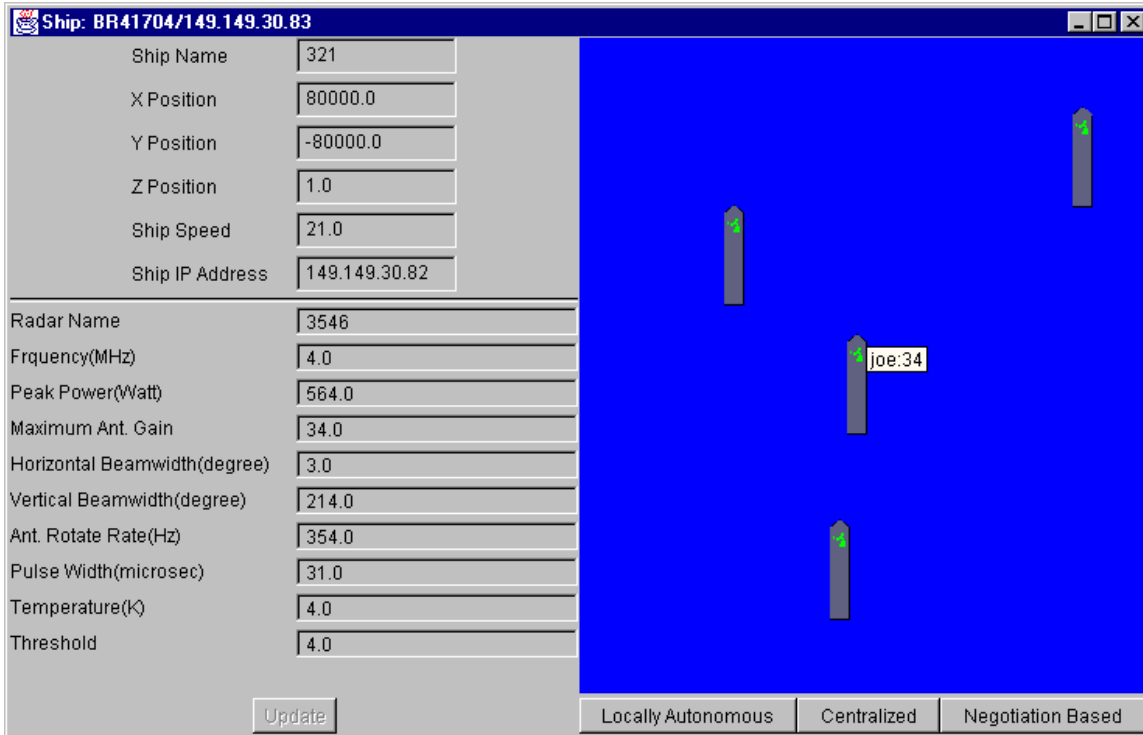
### **Higher-level Control Agent:**

The higher-level control agent is implemented using the Server, ClientThread and ServerGUI, Database, Inter-radar Interference resolution algorithm and Ship classes. The server class runs in an infinite loop ready to accept a client connection from any other control agent. A new thread is started for each client connection made to the control agent server, which is taken care of by the ClientThread class. This class also establishes the conversation between the control agents. The ServerGUI class pops up a window for each client connecting to it. The ServerGUI class pops up a window for each client connecting to it. The database class, as mentioned earlier, has information of radar units of all ships in the group. The ClientThread and Inter-radar interference resolution algorithm classes in the victim and source ship respectively, use this class for resolving interference problem. The schematic of the implementation of the higher level control agent is given below. The Inter-radar resolution algorithm class again runs in an infinite loop waiting for a flag from the lower level control agent informing it that there is an external interference. Once it gets the flag it takes information about the affected radar from the lower level control agent and looks up the database to detect the source of interference. Once it identifies that the source of interference is from a radar unit from another ship in the group it starts a client connection and a negotiation with the higher- level control agent of that ship. The ship class is the main class in the lower-level control agent is dealt with in the next section.

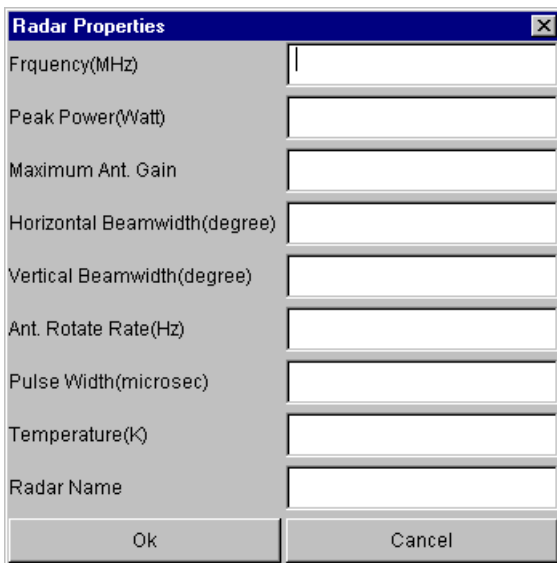


**Lower-level control Agent:**

As mentioned in Section II this class is responsible for checking whether a radar unit is experiencing a local or external interference. If the interference is local it resolves it appropriately, otherwise it sends a flag to the Inter-radar interference resolution class of the higher level control agent stating that a radar unit is having an external interference and passes the operating parameters of the affected radar. The lower level control agent is implemented using the Database, ShipCanvas, ShipImage, RadarImage, LwToolTips, ShipInfo, Radar and RadarInfoDlg classes. A screenshot of the implementation of the lower level control agent i.e. the Ship class is given below. The ShipCanvas and ShipImage classes are used for drawing the ships on the graphic user interface of the Ship class. The Radarimage class is used to draw the image of the radars on the individual ships respectively. Operating parameters of each individual radar unit of the local ship can be observed by pointing the mouse to a particular radar unit on the local ship. This is incorporated by the LwToolTips class.



The task of adding a new radar unit or initializing the operating parameters of a particular radar unit on a ship are taken care of by the Radar and RadarInfoDlg classes. A screenshot showing the radar information dialogue box that is used for adding or initializing radar units on a ship is given below.



#### IV. Conclusions

In this paper, the two-level agent architecture for resolving interference problems between ships with multiple radar units has been presented. Currently, the database handler has been implemented and the lower and higher-level control agents are being implemented. The main advantages of having such a architecture are the following:

1. Distinguishing interference problems within radar units on the same ship, or intra-radar interference, and interference between radar units on multiple ships, or inter-radar interference is useful is devising appropriate resolution schemes.
2. Maintaining a periodically updated database containing information about radar units on all ships within a group is useful in localizing much of the decisions made and hence reducing network traffic.
3. The two-level architecture helps in achieving a good balance between addressing requirements related to local and system goals. While the local goal of providing interference free operating frequencies to radar units on a given ship is uniquely addressed by the lower level control agent, the higher level control agent maintains responsibility for issues such as inter-group communication and coordination, and database consistency.

Future work will be focussed on the following issues: *i*) tuning the interference detection algorithm to take into account other issues caused by radar units on the same ship, and, *ii*) implementing negotiation based communication strategies between higher level control agents.

## V. References

1. H. Adnan, S. Ramaswamy, R. Macfadzean, "An Agent-based Distributed Simulation Environment for Frequency Assignment in Mobile Radar Units", *1998 Summer Computer Simulation Conference*, Reno, Nevada, July 1998, pp. 340-345.

