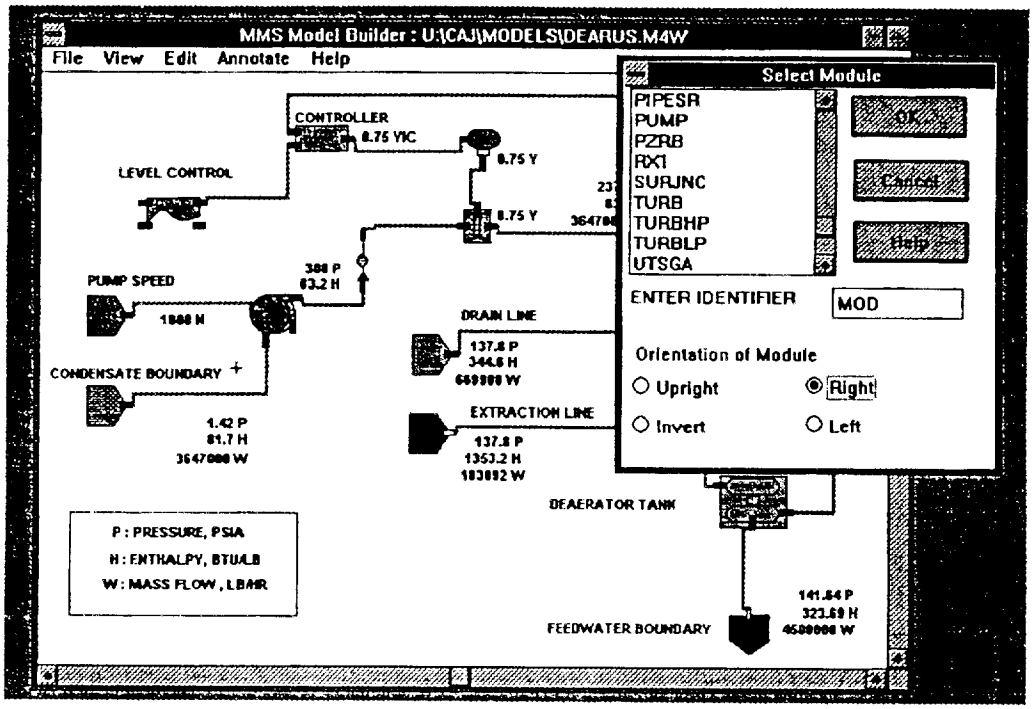

INTEGRATING REMOTE ACCESS AND CONTROL METHODS INTO SIMULATION MODELS WITH MMS

By

Charlie Alan Jones

B&W Nuclear Service Company
Engineering Services
Lynchburg, Virginia 24506

November 1993

Presented at the "Simulations, Modeling, and Training - 1993 EPRI International Conference"

Integrating Remote Access and Control Methods into Simulation Models with MMS

Charlie Alan Jones
B&W Nuclear Technologies
Engineering Services
Lynchburg, VA 24506

Abstract

Power plant simulations are invaluable for training, design, system optimization, and control hardware testing in the power generation industry. Obtaining optimum use of simulation models often requires a customized user interface that requires extensive changes to the model and occasionally requiring different simulation software. Fortunately, new operating systems are available today with features to ease the task of creating these interfaces.

Remote access and control methods, integrated into the Modular Modeling System (MMS), provide the user a way to interact with a model without having to modify its source code. Several development environments are now available that exploit these remote access services to make simulation models more useful and attractive.

The MMS Model Builder greatly simplifies development of MMS simulations. The Model Builder is a graphical interface used to construct schematic representations of a model by accessing modules from the MMS Library of power plant and control components. The Model Builder automatically generates text files for use in simulation development, and includes an array of user productivity features such as extensive on-line help, drag, pan, zoom, and cut and paste capabilities. Calculation of component parameters based on device data is presented.

Remote access implementations are depicted using examples. Multi-tasking and network-distributed implementations show the flexibility of remote access methods. Examples use different application development packages to demonstrate the user's ability to choose a development package that suits his application and programming experience.

Introduction

The MMS software package has undergone significant change in order to meet customer needs and to take advantage of new technologies. Downsizing of development projects to the desktop (PC) environment is a current trend in the industry and the MMS customer base. Limitations on the size and capabilities of desktop simulations continue to disappear as new benefits in cost and ease-of-use continue to emerge. These emerging technologies have provided extensive and, perhaps under appreciated, development opportunities.

Multi-tasking operating systems for the desktop have allowed developers using the MMS to reduce the effort required in building custom interfaces. Microsoft® Windows™ 3.1 and Windows NT™ 3.1 provide multiple and unique methods for connecting simulation models to custom user interfaces.

Inter-Process Communication (IPC) is the common industry term for the type of connection methods that will be discussed. Many IPC methods are available and the advantages and disadvantages of each will be considered.

The end result of this technology, as will be shown, is the tools necessary to build power plant simulations that can easily be extended with custom user interfaces. IPC also allows the developer to separate computationally intensive processes for increased simulation response.

Building MMS Models with the MMS Model Builder

Creating the Model Diagram

MMS is designed to provide engineers with high-level power plant components with which to build simulations. Figure 1 shows the MMS Model Builder and the Select Module dialogue box. Full zoom, pan, text and rectangle annotations, printing, block move, cut, copy, paste and on-line context sensitive help are supported in the interface[1]. Object-oriented programming methods were used to develop the MMS Model Builder using the C++ language[2].

Building a model requires the user to select the needed components and place them on the worksheet. Components are connected with the mouse to define the flow paths and control signals. Once modules are inserted on the worksheet, all the input parameters are available through a Module Data Input dialogue box. Figure 2 shows the dialogue box for the PUMP module. In order to create a simulation, modules must provide engineering parameters, such as heat transfer coefficients. Each module in the interface provides a routine that takes known plant input and produces the correct parameters. This process is known as Auto-parameterization.

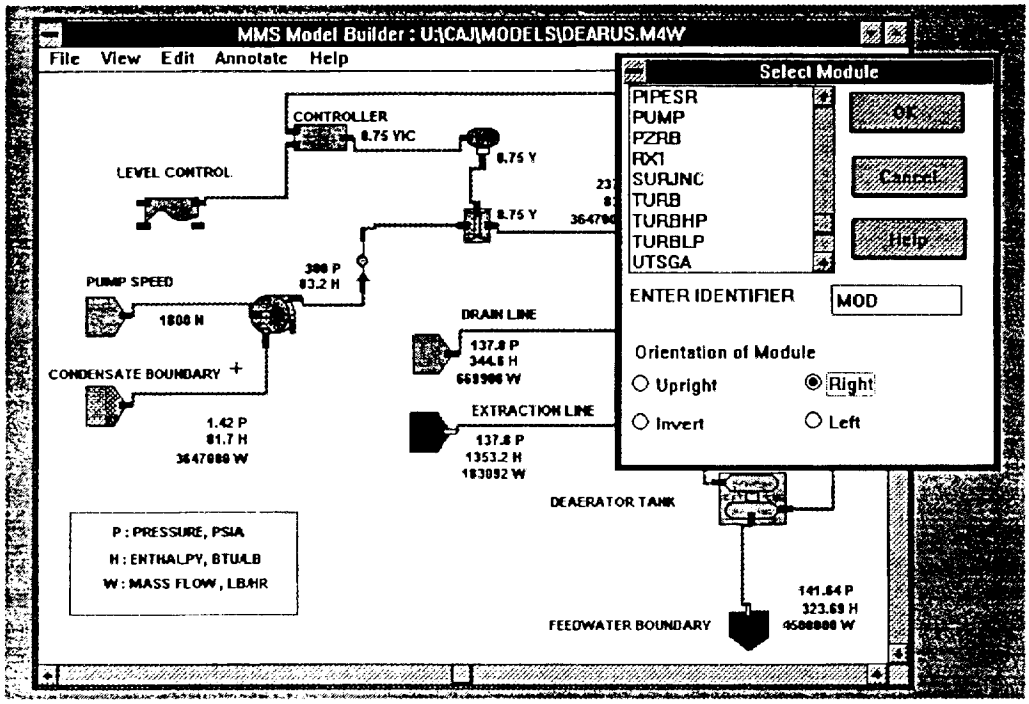

Figure 1
MMS Model Builder Interface and Select Module Dialogue Box

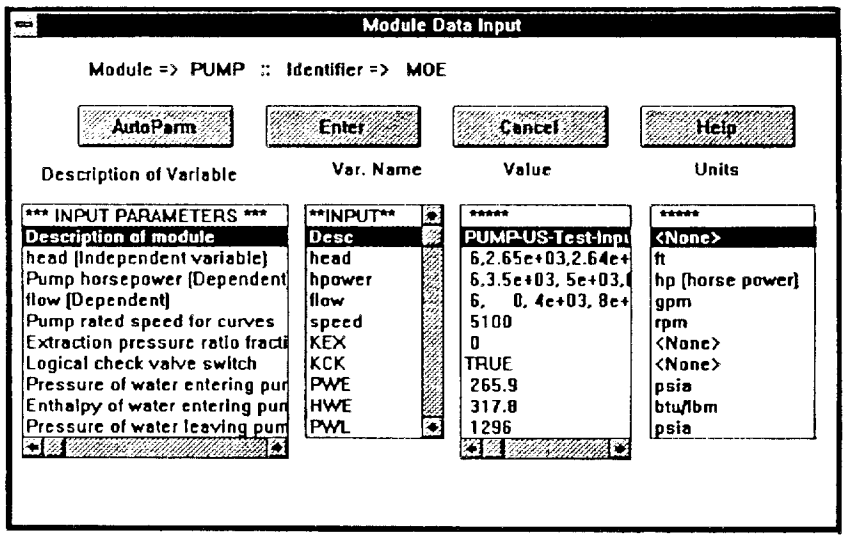

Figure 2
Module Date Input Dialogue

ACSL Source Code and Building the Executable

The Advanced Continuous Simulation Language (ACSL) is the simulation language used by the MMS[3]. The MMS Model Builder diagram and database are read by a source code generator to output the ACSL source code to a file. The ACSL/PC for Windows builder is then used to create the executable for the Microsoft® Windows™ platform[4]. ACSL supports a wide range of mini and mainframe computers. The source code generated by the MMS Model Builder can be used, without modification, on any machine for which ACSL is available.

In building an executable, ACSL converts the source code into a FORTRAN source code. The FORTRAN source code is compiled and linked with the ACSL runtime modules. MMS has a runtime library that must be linked with the executable as well.

Once the executable is built, the model can be exercised. Starting, stopping, restarting, setting of variable values and plotting can all be accomplished from the ACSL interface. During the development of a simulation, models are primarily exercised through the ACSL runtime interface.

Building Custom Interfaces for MMS Simulations

Previous Methods

Static linking of an interface subroutine has been the method of connecting user-designed graphical interfaces to MMS simulations. ACSL supports the introduction of a FORTRAN subroutine directly into the ACSL source code. Placing the subroutine in the DYNAMIC section of the ACSL source code forces the subroutine to be called at each communication interval. (Information on the ACSL program structure and communication interval can be found in the *ACSL Reference Manual.*) All of the parameters necessary for the interface are passed in the argument list of the subroutine.

The resulting FORTRAN source code is then compiled and linked with user-written routines and a graphics library. In this way the object code for the interface becomes part of a single executable file.

Advantages of IPC Methods

IPC methods have many advantages over static linking. Microsoft® Windows™ 3.1 and Windows NT™ 3.1 support a wide range of IPC methods. This allows the designer to select the method that best fits a particular situation. In general, all IPC methods separate the interface development into a task that is independent of the simulation.

Separation of the interface frees the designer from the chore of rebuilding (ACSL translating, FORTRAN compile and linking) the simulation model every time the interface is modified. With large models this can save a significant amount of man-hours. Program maintenance is simplified because the program tasks are separately constructed programs. Operating system support allows many of the IPC methods to support executing the simulation and interface on separate machines

over a network, or as separate processes on a multi-processor computer. Different interface programs (one for design, another for training) can use the same simulation without change to the simulation source code. Field changes to a delivered product are simplified by replacing only the module that needs to be changed.

IPC Methods and Their Use in MMS Simulations

A wide range of IPC methods are provided by Microsoft® Windows™ to meet the entire spectrum of possible application design needs. Not all the methods available are suitable for use with the MMS simulations. Table 1 categorizes the capabilities and suitability of each. Detailed explanation of each method is available in reference 5.

In Table 1, multi-platform support is applied to IPC methods that support connection to operating systems other than Microsoft® Windows™. Loose binding provides connection to other tasks by process name or by server query. Programming complexity is based on the author's experience in creating MMS simulations and the current development tools available.

IPC Method	Multi-Platform Support	Network Support	Suitability for Simulation	Programming Complexity	Binding Method	Speed
Shared Memory	No	No	Medium	Medium	Tight	Fast
Anonymous Pipes	No	No	Very Low	Low	Tight	Fast
Named Pipes	Yes	Yes	High	Medium	Loose	Medium
Mail Slots	Yes	Yes	Limited	Medium	Loose	Slow
DDE	No	Yes	High	Low	Loose	Medium
Object Linking and Embedding	No	Yes	Medium	Very High	Loose	Medium
Shared DLL	No	No	High	Medium	Tight	Fast
Remote Procedure Call	Yes	Yes	High	Medium	Loose	Medium - Fast
Net BIOS or other Direct Network Calls	Yes	Yes	High	High	Tight	Medium - Fast

Table 1.
IPC Methods under Microsoft® Windows™ and Windows NT™

Methods Suitable for MMS Simulations

Simulations run exclusively on a single machine may take advantage of Shared Memory or Shared Dynamic Linked Libraries (DLL). Both of these methods provide the fastest method of communicating data between processes. Poor program maintenance and poor separation of the source code programming for the two processes are major disadvantages. Converting the simulation to use any of the other IPC methods would be difficult. Direct network calls permit network access but suffer from the same code maintenance problems and difficulties in converting the code.

Named Pipes provides a very general method for establishing a connection between separate processes. Named Pipe servers can provide their names to allow clients to determine to which service they need to connect. Simulations that support two or more interfaces and have low to medium data transfer needs are good candidates for Named Pipes.

Dynamic Data Exchange (DDE) is a very general, robust and rich method for IPC under Microsoft® Windows™. The primary shortcoming is the lack of multi-platform support. DDE has several advantages not offered by any other method. The specific data to be communicated and communication interval can be configured at runtime. DDE supports event-triggered communications; the requested variable's value is only communicated if the value is changed. Many high-level development tools support DDE. A program can serve as both a DDE server and client. Servers can make their names available for multiple interface support.

Remote Procedure Call (RPC) has many of the advantages of DDE. Additionally, it provides support for other operating systems. RPC has more flexibility in how services are connected at runtime and provides access across many types of networks. However, RPC is less versatile in changing the availability of variables during runtime.

ACSL IPC Capabilities under Microsoft® Windows™

The ACSL/PC for Windows runtime interface supports the DDE IPC method. DDE provides a high degree of compatibility with other Microsoft® Windows™ applications. Simulation variables can be connected to a custom interface with a minimal amount of effort. Spreadsheets, word processors and drawing programs can be configured as an interface for a MMS simulation program.

Example Applications

Selecting IPC Methods

Two example interfaces are developed using the DDE and RPC methods. DDE was selected because of the large number of high-level development tools currently on the market. This puts the program development of a simulation interface within the skill level of a large number of computer users.

RPC was selected because this technology seems to present the best balance between programming ease and the performance necessary to complete large simulations. Simulations requiring multiple display screens and operator stations would be best served by the RPC technology.

Deaerator DDE Custom Interface

Figure 1 showed the development of a deaerator level control simulation with the MMS Model Builder. This model is the basis for the DDE custom interface. Microsoft® Visual Basic™ was used to develop the interface. All simulation variables are available to the interface using the DDE method. DDE also supports sending commands to the ACSL command line.

DDE provides the greatest independence between simulation development and interface development. No changes to the source code are required to provide for DDE conversations. Simulation developers can produce models for resale to users who can provide their own interface without access to the simulation source code.

Figure 3 shows the deaerator example. The text and trend chart are constantly updated as the model runs. Deaerator level setpoint can be controlled by the scroll bar while the model executes. Any number of controls can be placed on the interface. Visual Basic™ can support multiple forms (windows) connected to the same model. Several different interface programs can access the model at the same time as well. The DDE model and interface must be run under Microsoft® Windows™ 3.1.

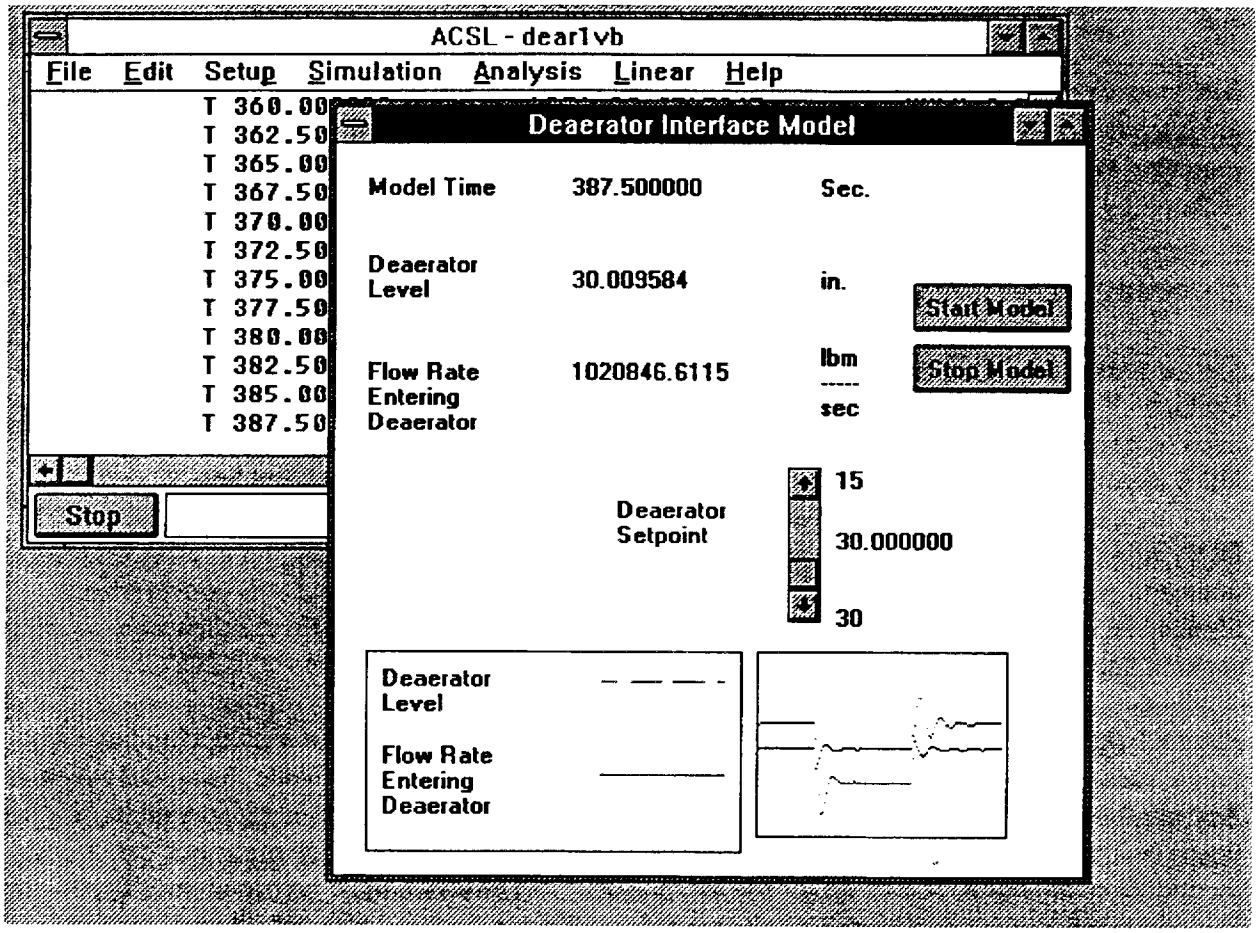

Figure 3.
DDE Custom Interface and the ACSL/PC for Windows Runtime Window

RPC Example Interface

The RPC model and interface were built and tested under Windows NT™ 3.1. The client and server models are written in C. All source code projects were built with the 32-bit version of the Visual C++™ compiler. Simulations run on multi-processor machines running Windows NT™ 3.1 can take advantage of the multiple processors without source code changes. This would not be true if the simulation were limited to a single non-threaded process (static linking).

The creation of the source code for the client and the server is greatly simplified by use of the Microsoft® Interface Definition Language (MIDL) compiler[6]. The MIDL compiler accepts as input two files that describe the subroutine parameters and network configuration. Output of the compiler is C source code necessary to provide the RPC calls needed for the connection. MIDL-produced code was used in both the client and server applications in this example. A MMS application would need to have a subroutine in the source code to provide connection to a RPC server.

RPC allows for connection to different interfaces that can be determined at runtime. Multiple client and server configurations are possible. The RPC services in the example are started and stopped from the client application menu. If the server has not been started, the client prompts the user to start a compatible RPC server.

Figure 4 shows the RPC example. The transient is started from the client menu. Calculated values are displayed in both the server console and in the client controls during execution. The color of the boiler drum and the turbine are animated to reflect the current state of the transient.

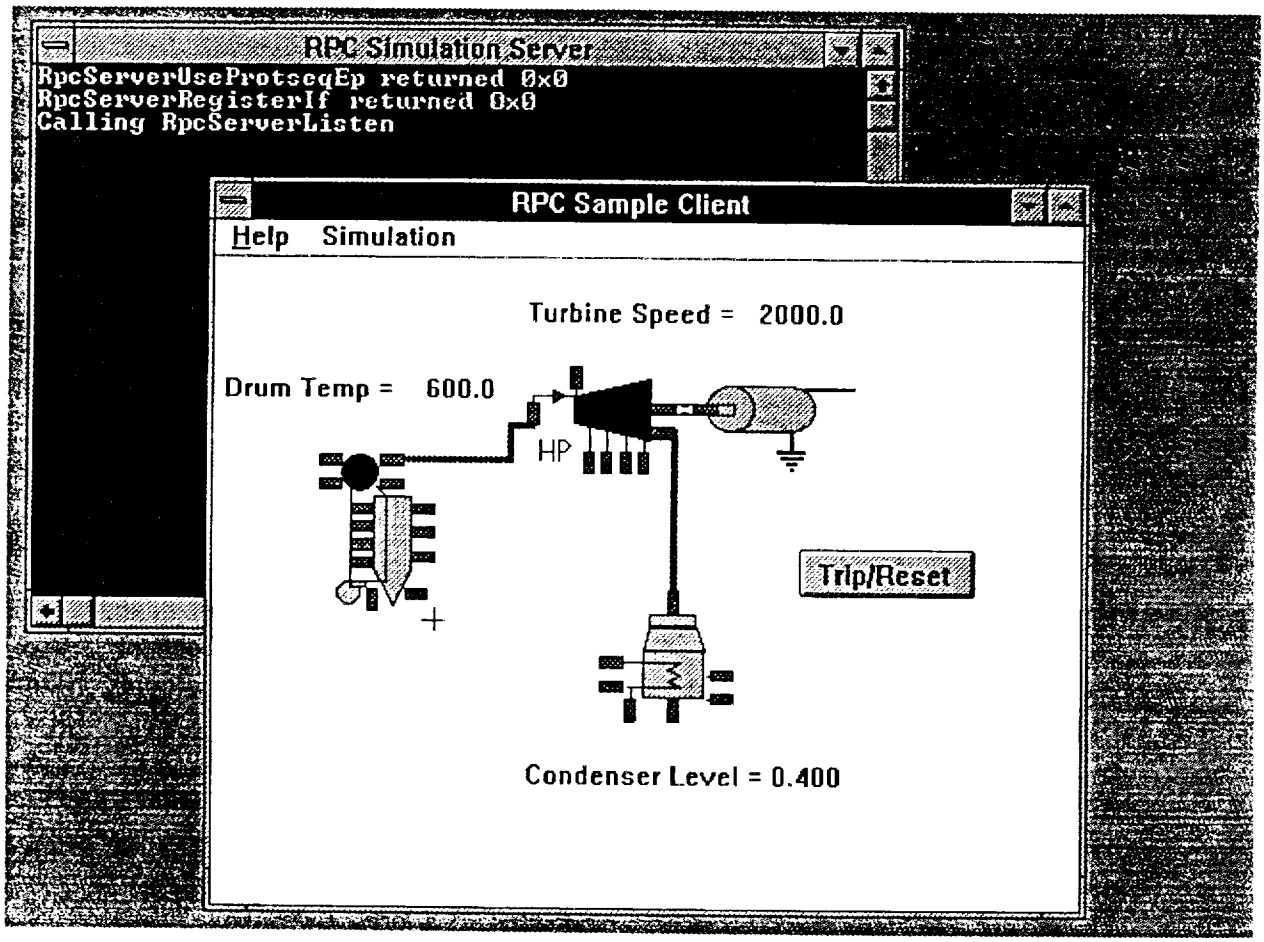

Figure 4.
RPC Client and Server Applications

Conclusions

MMS models can be easily constructed using the MMS Model Builder program. These models are converted to ACSL source code from which an executable is built. Models are tested using the ACSL runtime interface. Custom interfaces can be added to the simulations using staticly linked or IPC methods.

IPC methods can provide a wide range of benefits to the simulation developer. These methods can be used to modularize the code; increase runtime efficiency; provide network support; and provide for multiple client-server configurations. These advantages come at very small overhead in programming effort. Two example client-server simulations are presented to demonstrate the benefits of the selected IPC methods.

DDE can provide complete control over a simulation without any change to the simulation's source code. The DDE example shows the ease with which an interface can be built using DDE. Interactive control of the model while the model is running provided in this example.

RPC methods and animation are demonstrated with the second application. This example also shows how console applications can communicate with graphically based applications under Windows NT™.

Using IPC methods with MMS simulations provides simulation developers with a definitive set of tools for overcoming the traditional limitations of large simulations. This includes separation of programming tasks and network support.

REFERENCES

1. *MMS Model Builder User's Guide*. Lynchburg, VA.: B&W Nuclear Technologies, March 1993.

2. C.A. Jones May 1992. *An Object-Oriented Approach to Graphical Interaction with the Modular Modeling System*, 8th Power Plant Dynamics, Control, & Testing Symposium, Knoxville.

3. Mitchell and Gauthier Associates, Inc. 1991. *ACSL-Advanced Continuous Simulation Language-Reference Manual*, Mitchell and Gauthier Associates, Inc., Concord, Mass.

4. Mitchell and Gauthier Associates, Inc. 1993. *ACSL/PC for Windows - Installation and How to Use*, Mitchell and Gauthier Associates, Inc., Concord, Mass.

5. Microsoft® Win32™ Software Development Kit for Windows NT™, *Programmer's Reference, Volume 2: Systems Services, Multimedia, Extensions, and Application Notes,*

1993. "Chapter 85 - Interprocess Communications Options in Win32 Applications", from Microsoft® Developers Network CD-ROM, Disk 4, Summer 1993.

6. Microsoft® Win32™ Software Development Kit for Windows NT™, *RPC Programmer's Guide and Reference*, 1993. "Chapter 10 MIDL Language Reference", from Microsoft® Developers Network CD-ROM, Disk 4, Summer 1993.

Definition of Terms

ACSL - Advanced Continuous Simulation Language from Mitchell and Gauthier Associates, Inc. The simulation language used by MMS.

DDE - Dynamic Data Exchange. DDE is a service provided by Microsoft® Windows™ to allow applications to communicate data.

DLL - Dynamic Linked Library. DLLs provide for subroutine calls to programs in much the same way as conventional libraries. The difference is DLLs is the operating system loads the library and provides the subroutine addresses at runtime.

IPC - Interprocess Communication. Any software that allows programs to exchange data.

MIDL - Microsoft® Interface Definition Language. A source code used by the MIDL compiler to create C source code files. These C source code files can then be used to provide RPC services by compiling and linking them with an application.

Net BIOS - Network Basic Input/Output System. A programming interface for exchanging data between network applications.

RPC - Remote Procedure Call. A programming interface that permits subroutines to be located and called at runtime.