MACSIM: AN AGENT ORIENTED SIMULATION CODE FOR THE MNR

P.Gérard, Wm. J. Garland and W. F. S. Poehlman*

Department of Engineering Physics, * Department of Computer Science & Systems
McMaster University
Hamilton, Ontario
Canada, L8S 4M1

As reported in [GAR93], one relevant issue confronting the operator of a nuclear power plant is information and task overload. The goal of improving the environment for operators and technical support staff is accomplished by providing tools and techniques which reduce staff involvement in low level tasks and assist in information and knowledge manipulation so that high level tasks can be performed more efficiently. Past work by the McMaster University Performance Support System Group (PSSG) focussed on the OPerator/User Support project (OPUS), an operator companion for the Secondary Side Chemistry System at Pt. Lepreau. Therein, the feasibility of a distributed, loosely coupled network of computer programs, organized as a society of computing agents (manager, supervisors, technicians) was proven. Current work is aimed at implementing this strategy for the McMaster University Nuclear Research Reactor (MNR), a 2 MWth swimming pool reactor.

1. Introduction

MACSIM is, in essence, a PC-based (Personal Computer) multitasking environment which partitions the different tasks involved in the simulation of the reactor systems. The different agents act either as manager, supervisor or technician, relaying all the data to a single agent, the "blackboard". This enables the simultaneous execution of real time data acquisition, neutron flux simulation, thermal hydraulics

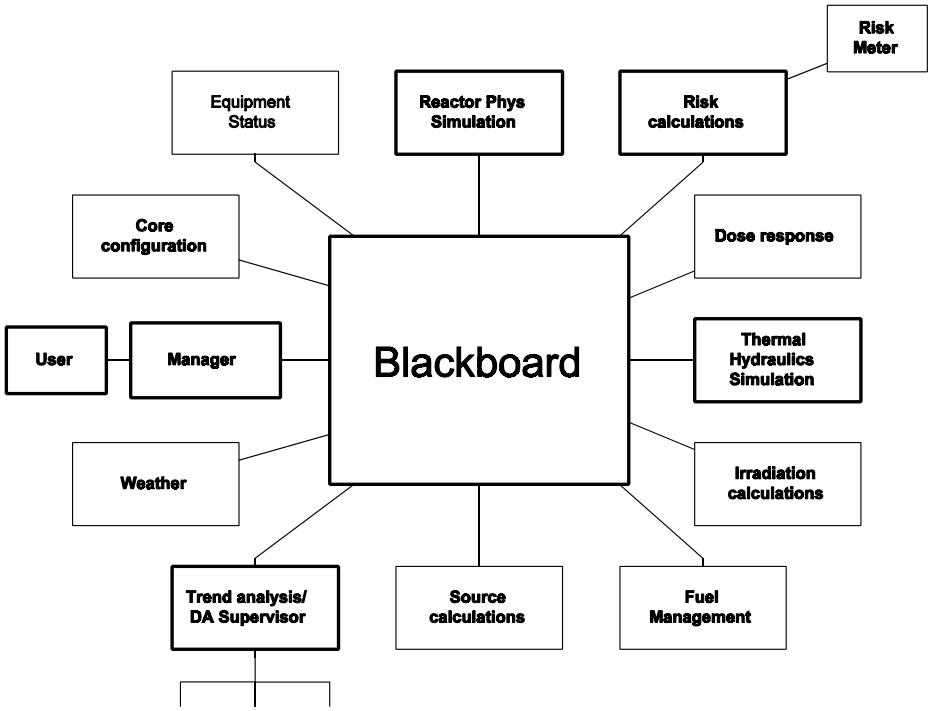

Figure 1 Conceptual Layout of agents in MACSIM

simulation and other codes as well as subsequent exchange of data also in real time. Figure 1 illustrates the conceptual layout of the agents. The agents in bold are under development; the rest are planned for the future.

OPUS used X_Windows™ and TCP/IP on a Ethernet based LAN to provide the networking functionality. MACSIM utilises LabVIEW [1] for data acquisition, the Graphical User Interface, and the network communication (via TCP/IP and Ethernet). Figure 2 illustrates the overall schema. At this time, the blackboard is a spreadsheet (Quattro Pro) file that can be dynamically accessed to read or write information in a given block of cells.

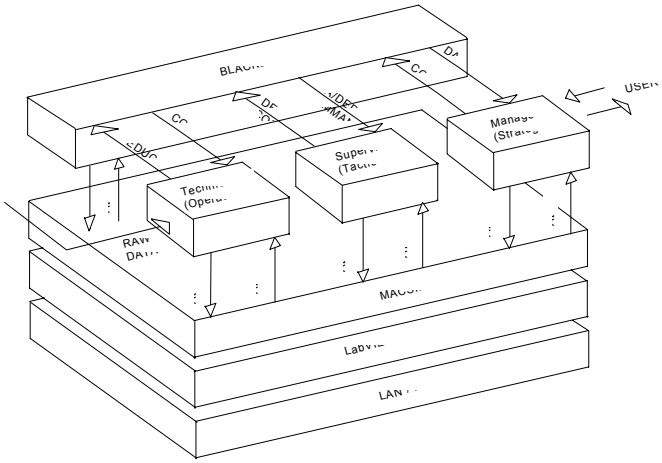

Figure 2 Overall schema

MACSIM is designed to be used as an analysis tool to assist the technical support staff and operators of the MNR. MACSIM could be used to simulate procedures (core shuffling) beforehand, to monitor the core in real time and compare to simulations, to provide a real-time assessment of facility safety/risk status, or to simulate planned experiments.

This paper is intended to provide an introduction to the concept and structure of MACSIM. Section 2 consists of a brief introduction to LabVIEW. Section 3 describes the general structure of MACSIM. Section 4 illustrates through an example the tools used and section 5 contains typical results produced by MACSIM. Finally, section 6 underlines the performance issues.

2 LabVIEW overview

LabVIEW is a PC-based (Personal Computer) program development application which uses the graphical programming language G. LabVIEW programs are called Virtual Instruments (VI). VIs have the same properties as C functions in that they can be called by other VIs and data can be passed between levels. LabVIEW contains many libraries of built-in VIs for various tasks. LabVIEW user interface resources allow the user to easily interact with MACSIM by the click of a mouse using flexible displays and controls. Furthermore, the computational results are immediately available to the user in graphical and numerical form. The controls and displays are easily created and are linked to the block diagram which contains the graphical source code.

---

[1] LabVIEW is a trademark of National Instruments Corporation. Copyright 1992, 1996.

The calculations are performed using C functions located in MS Window-based Dynamic Linked Libraries (DLL). The data is passed from LabVIEW to the C functions using Call Library Function Nodes. The DLLs are programmed in C much like any other library with the only exception is that the Window's function LibMain and WEP must be present and the function must be declared as extern "C". C functions are executed synchronously, which means that all other tasks are suspended on the given computer.

The communication protocols Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Dynamic Data Exchange (DDE) are available in LabVIEW for IBM compatible personal computers. For communications between computers, MACSIM makes use of the built-in TCP VIs. TCP is favoured over UDP because this protocol ensures reliable transmission. For communications between various Windows application (ie: LabVIEW and Quattro Pro), MACSIM uses the DDE VIs. Although not exploited at this time, LabVIEW can also be used for real-time data acquisition.

3 MACSIM

3.1 Blackboard

The blackboard philosophy is used to facilitate the logistics of data exchange between codes and in a latter stage, ease the integration of real time data acquisition. Instead of sending specific information to a given code and having to take into account the computing time of each code, all the latest relevant data are 'posted' on the blackboard and are continuously available to any other module. Furthermore, by registering and unregistering computers on the blackboard, the module dispatcher has a better idea of the resources available.

Using built in DDE VIs, data contained in a spreadsheet file can easily be accessed by specifying the cells' block name and requesting its content. Conversely, the cells' content can also be modified. In some instances, to reduce the communication time, data are grouped together and transferred as such.

Unfortunately, there is a limit on the size of data that can be transferred by DDE. When this limit is exceeded, the computer has a tendency to crash. The flux at each grid point for a given energy level exceeds this limit, as well as the concentrations of the tracked isotopes at each grid size. These data must therefore be saved to the hard disk and transferred by TCP connection when required.

3.2 MACSIM's structure

All communications are based on a client/server model. A Client VI initiates a Open Connection command for a given IP address and port# (figure 3). A Server is a VI that is continuously in a listening mode, waiting for an Open Connection command from a Client on a remote computer on the specified port (figure 4).

A library of Client/Server pairs was created to handle various tasks (VI control, DDE/TCP, read/write files, execute...). Each pair of VIs has a different port # such that different communication tasks can be executed simultaneously on a given computer. Although, the message structure varies from one pair to another, in all cases all the information is bundled together and converted to a string. Commands are usually passed in the form of integers (0 ,1 ,2 ,3 ...).

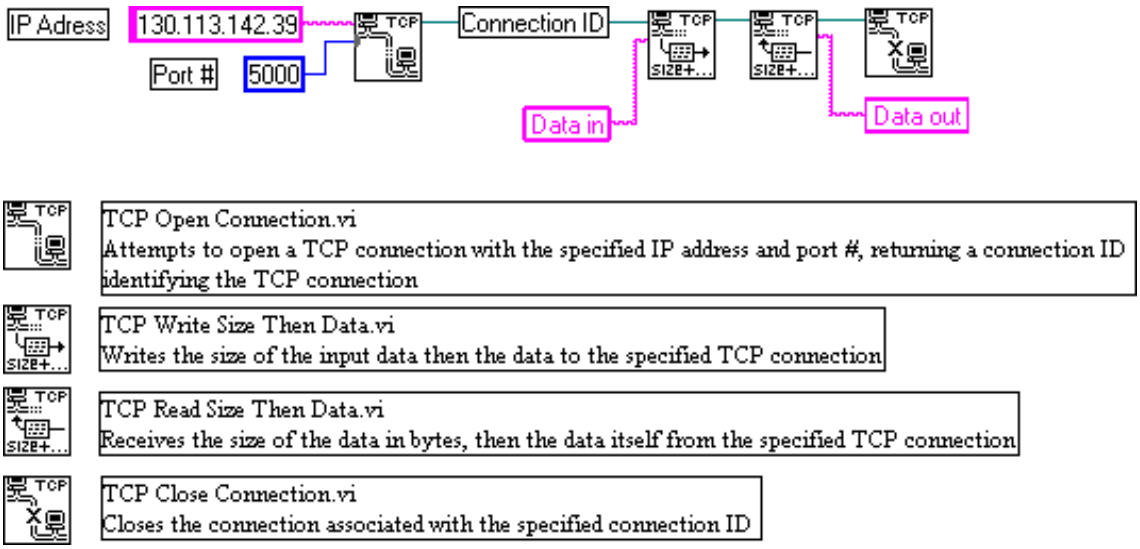

Figure 3 Client VI

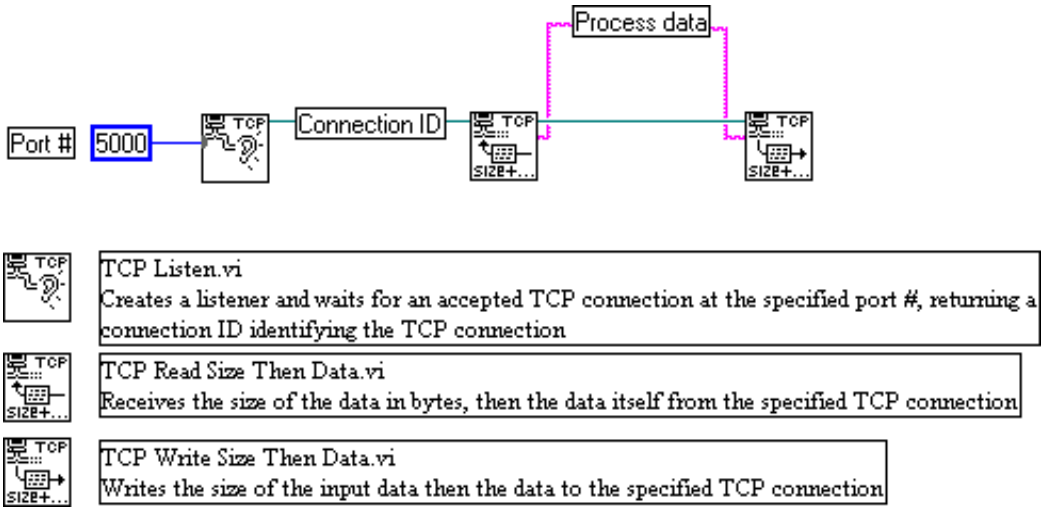

Figure 4 Server VI

Although client/server pair have specific tasks, they are built to be as general as possible. For instance, there is a Client/Server pair for the remote control of VIs (figure 5). This pair allows the launching/closing of VIs on remote computers. The message consists of a command (0 - Close, 1- Standby, 2- Execute, 3- Stop execution and close), the VI name and path and other data concerning the window state of the VI. This is the Master Client/Server pair as it can launch any other Server and Client VIs when required. In other words, the Master Server is all that is required on a remote computer to launch any LabVIEW VI.

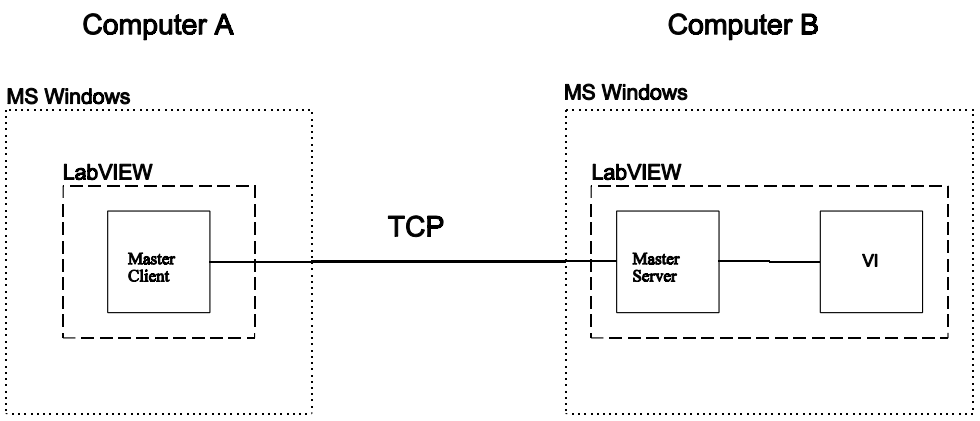

Figure 5 Master Client/Server VI pair

The blackboard is accessed remotely by a DDE/TCP Client through a DDE/TCP Server that, given the proper information, can access data in a Quattro Pro spreadsheet file (or any other Window's application that accepts DDE connections). This is illustrated in figure 6. The message consists of the topic (usually file name), the service (QPW, MATLAB), the command (0 - Request, 1 - Poke and 2 - Execute), the item requested or the command to be executed and data for the Poke command.

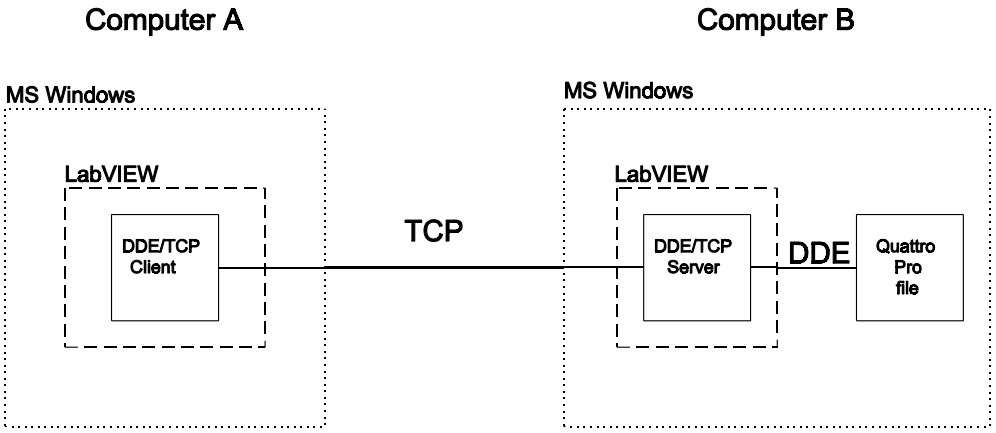

Figure 6 DDE/TCP Client/Server VI pair

The user does not have access to any client/server VI. Client/server VIs are used by the three types of agents; manager, supervisor and technician. The user solely interacts with a manager agent that relays information back and forth from the blackboard using client/server VIs (figure 7). The manager also establishes the general strategy to solve the specified problem, launching the supervisor agents on the available computers on the basis of free computer resources. Supervisors are launched using a supervisor dispatcher. Ideally, any computer should be able to execute the various tasks. Moreover, any user station should be able to display all the relevant data contained in the blackboard at any time.

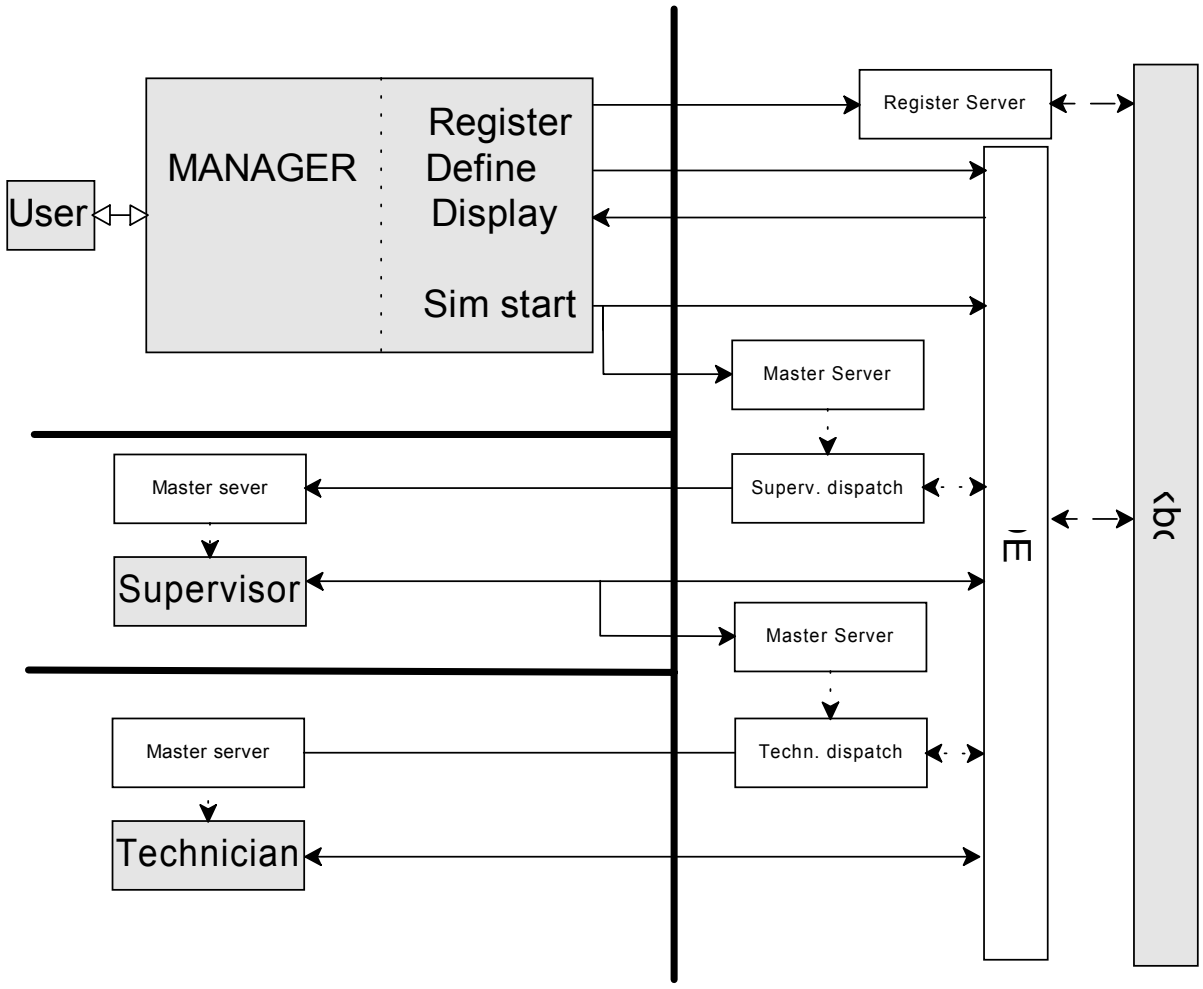

Figure 7 MACSIM's Structure

A supervisor agent coordinates the different tasks and results, sending commands to the various technicians through the blackboard. Once the technicians are working, the supervisors monitor the progress, verifying that no technicians are stalled and thus delaying the other technicians. Finally, for each task, there is a technician module that, once launched, periodically probes the blackboard (frequency specified by the user) for commands. In some instances, the technician will probe the blackboard once the current task is completed. These are known as proactive technicians. Once a task is completed the technician sends the results and a status message to the blackboard, indicating to the supervisor that it is now available.

The communications between the agents and the blackboard are asynchronous since communications are on a 'need to know only' basis.

4 Present status

Although MACSIM is still in the developmental stage, some modules are already functional and can be

use to illustrate and test the general structure of MACSIM. Simple reactor physics and thermal hydraulic simulation codes have been developed for the MNR (figure 8). Data acquisition is currently not real-time.

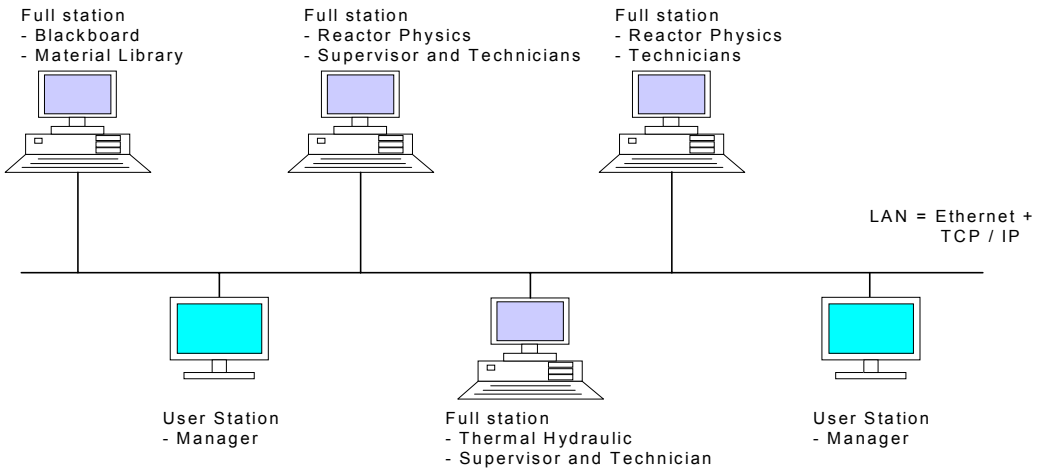

Figure 8 Present MACSIM status

Off-line Trend Analysis development is underway as is the development of the event and fault trees required by the risk agent.

Through the blackboard, a feedback loop is created between the reactor physics code and the thermal hydraulics code; the flux is used to determine the energy transferred to the coolant and the fuel temperature affects the cross-sections. The following sections gives an overview of the different modules and how they fit in the overall structure.

4.1.1 Reactor Physics Code

MACSIM uses a multigroup 3D diffusion semi-implicit code to solve for the neutron flux in the core. The core is divided in cells of homogeneous composition (FUEL, MOD, BER...). The algorithm uses difference equations that take into account heterogeneous core and varying spatial increments. The seven point difference equations are solved using a Gauss-Seidel iterative scheme.

The various cross-sections are divided into fixed and moveable cross-sections. This is to allow for different control rod insertions. The cross-section is also corrected for the presence of Xenon and Iodine in the cell, the depletion of U^{235} and U^{238} and the build up of Pu^{239}.

In addition to the flux calculations, there are other smaller reactor physics modules; the ‘poison’ module calculates the poison concentration as a function of time, the ‘depletion’ module calculates the depletion/build up of fissile materials and the ‘controller’ module act as the regulating rod controller and scram safety system by varying the control rod insertion. These modules will be executed at different frequency as they simulate phenomena occurring on different time scales.

4.1.2 Material Library

The material library is contained in a Quattro Pro file. At this time, MACSIM uses a four group material library generated from a WIMS cell calculation, although it is not strictly limited to four. For each material, there is a page containing the microscopic cross-sections and atomic density, from which the

macroscopic cross-sections are calculated.

Lattice cell cross-sections are automatically calculated when the material content is entered in a specific block in a "query" page. Using a DDE/TCP Client/Server pair, a technician located on a different computer can obtain the lattice cell cross-sections for a given material content. The material library does not have to be located on the same computer as the blackboard file and the list of available material can be requested, also using a DDE/TCP Client/Server pair.

4.1.3 Thermal hydraulic Code

The MNR primary heat transport system (figure 9) is comprised of two open air tanks; the pool, the hold up tank, a pump, a heat exchanger and connecting piping and associated valves. The water flows from the pool, through the core, to the hold up tank due to a head difference of about 8 m. Water is pumped from the hold up tank, through a heat exchanger back to the pool. The steady-flow energy equation is used to

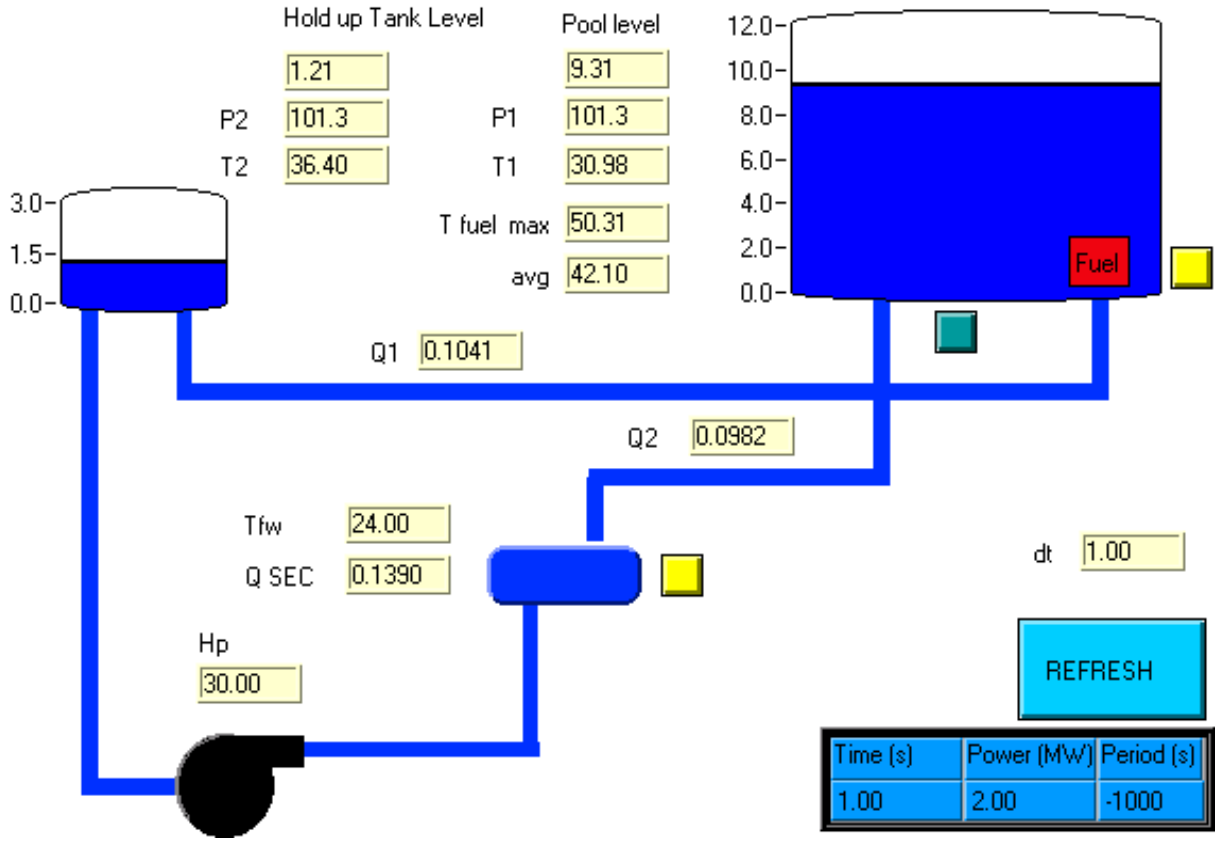

Figure 9 Heat transport system

calculate the flows from one tank to another.

For the heat exchange between the core and the coolant, and the primary and secondary side in the heat exchanger, the convective heat-transfer surfaces are partitioned and Newton's law of cooling is used. In

the fuel plates, the axial heat conduction is also calculated.

4.2 Manager

The manager at this point is composed of a series of subVI used to register the computer on the blackboard, define and enter data in the blackboard, start a simulation and display the results. The subVIs directly 'contact' the blackboard for the necessary exchange of data.

When a new computer is registered, its IP address is entered in the blackboard as well as its free system resources. This is used to dispatch the various modules once a simulation is initiated. Before a simulation, the user can define up to 10 cells, the core grid, the dimensions of the fuel plates and water channels, the controller parameters, the initial control rods insertions and the initial thermal hydraulics conditions. After the user has defined all the simulation parameters, the VI "Define simulation" decides on the basis of free system resources on which computer the supervisors must be launched. During the simulation, as well as at the end of it, there are several VIs created for the display of the results. Matlab™ is used to display some results in the form of surface plots.

4.3 Supervisors

At this time there are only two supervisors: one to coordinate the reactor physics technicians and one to coordinate the thermal hydraulic technician. Through the manager, the user specifies the type of problem to be solved. The manager relays the information to the blackboard which initiates the supervisors for reactor physics and thermal hydraulics on the available computers. The supervisor, from the information posted on the blackboard decides which technician to use (Flux, Controller, Poisons, Depletion...), at which frequency and which computer. Through a DDE/TCP client/server pair, the supervisor posts commands in the corresponding technician command fields. The supervisor also monitors the technician and reports the progress to the manager. Once completed, the supervisor advises the manager and shuts down the technicians to avoid cluttering up the computers needlessly.

4.3.1 Thermal hydraulic Supervisor

The thermal hydraulic supervisor main task is to monitor the power. When the power variations exceeds a set point, the supervisor orders the thermal hydraulic technician to execute the code. This approach is used to minimise the number of calculations required.

4.3.2 Reactor Physics Supervisor

The reactor physics supervisor has a more intricate task since it has several technicians under it that are executed at very different time steps. Depending on the type of problem, flux calculations may be performed at every time step or steady state calculations may be performed at larger time steps. For example, for simple burn up calculations, only steady state calculations may be performed at large time intervals during which the depletion and build of fissile material is calculated. On the other hand, for short transients, for example the analysis of the safety

system response, transient flux calculations must be performed at small time steps and may include xenon poisoning calculations at a larger time step.

If the flux calculations delay the remaining technicians, the supervisor will also delay the commands to the other technicians to avoid needless calculations. When the total transient time entered by the user has elapsed (in simulation time), the supervisor shuts down the technicians before shutting itself down.

4.4 Technicians

The technicians are basically composed of a DDE/TCP client and a call library function node to essentially transfer data from the blackboard to the C function and back to the blackboard. At a frequency specified by the user, the technician looks up its command field in the blackboard and updates the status field.

5 Simulation results

Since we are focussing on the general structure of MACSIM and the logistics of the data/command transfer at this point in time, the accuracy of the simulation codes are not our main concern. The cross-sections used in the reactor physics module and some correlations used in the thermal hydraulics module require further analysis. However, some of the results obtained in a typical simulation are shown in this section to demonstrate the type of results MACSIM generates.

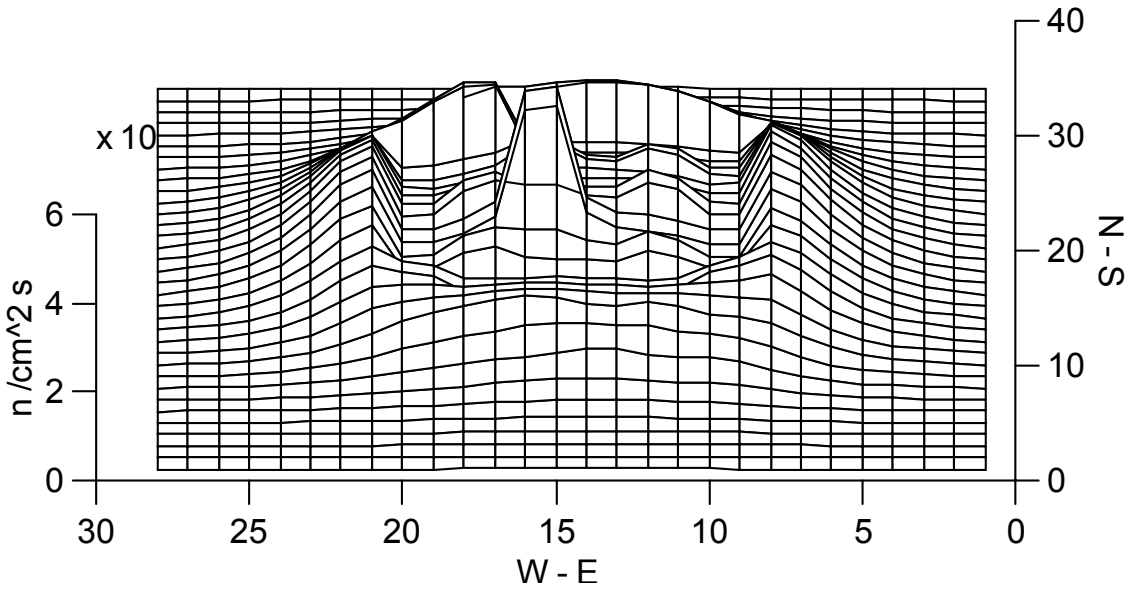

Figure 10 Flux map

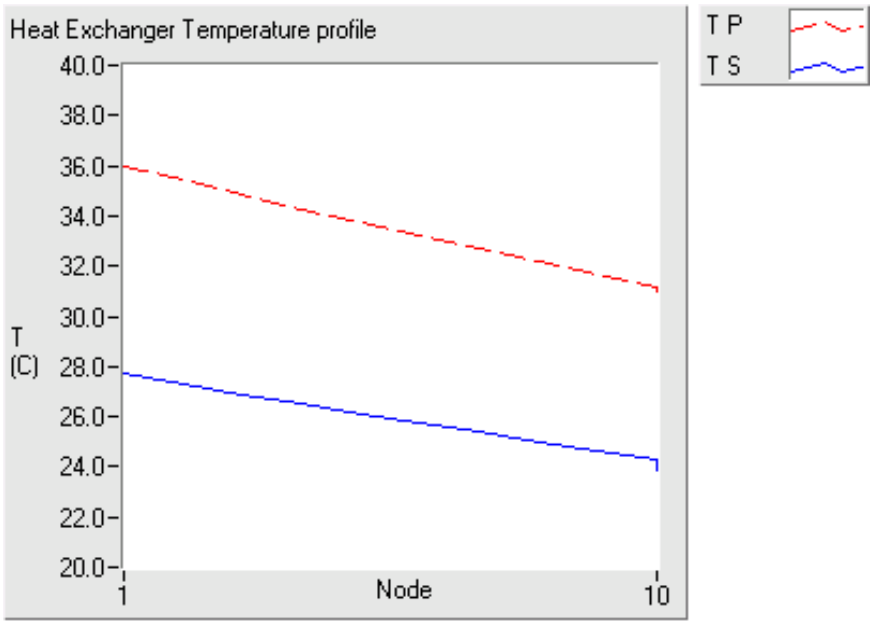

Figure 11 Heat exchanger temperature profile

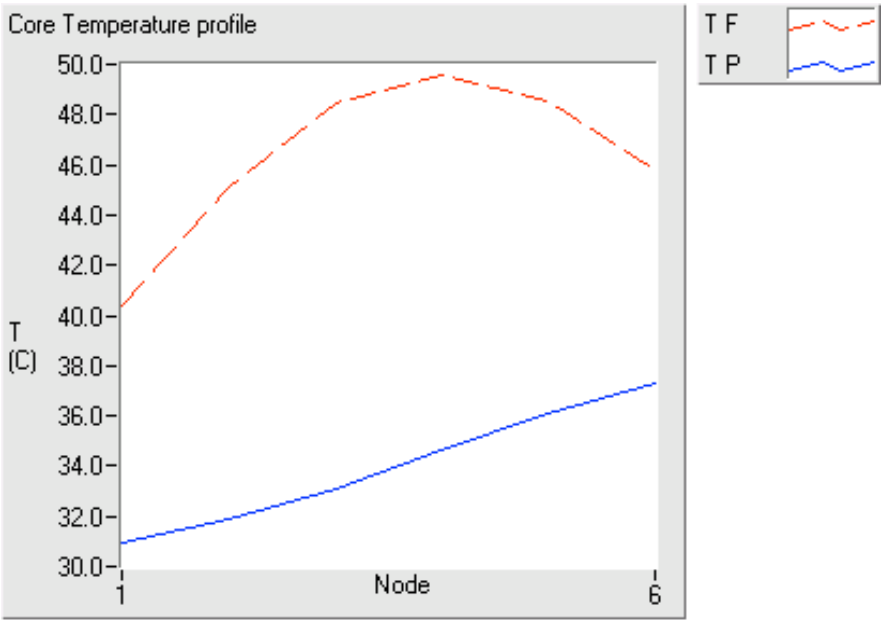

Figure 12 Core temperature profile of hottest bundle

Figure 10 shows a flux map of the mid section of the core (level 5 of 10) of the thermal neutrons (group 4 out of 4) at 100 % power. Using such plots allows the user to quickly identify hot spots in the core and effects such as the reflector on the thermal group. The large peak represents the irradiation site while the five humps represent the five out of six fuel cells (the sixth hump is hidden behind the large peak) in the centre of which control rods can be inserted. The flux seems to be in agreement with reactor data for 100% power. Similar plots of the concentration of the delayed precursors and the tracked isotopes are also available. The user is also provided with a chart of the power, reactor period and flows.

The thermal hydraulic results are summarised in figure 9 and the temperature profiles in the core (figure 11) and heat exchanger (figure 12). The core temperature profile shown in figure 11 is for the hottest channel in the core. Here again, the preliminary results seem to be in agreement with available data and other thermal hydraulics codes.

6 Preliminary performance report

In essence, a LabVIEW based distributed operating system was created for MACSIM. Distributed operating system have long been recognised for their significant advantages: resources sharing, improved reliability and availability and modular expendability. However, there are inevitably difficulties and penalties associated with the distributed approach.

Currently, one of the main concerns is the time penalty. The passing of commands and data between agents inevitably introduces a time penalty that is not present in a single module code. Unless there are other over-riding benefits, the administrative time, defined as the time required to

transfer data back and forth from the blackboard, must be relatively smaller than the computing time to justify the distributed approach. Furthermore, the administrative time also imposes a constraint on the time scale of the simulation; the simulation time step must be greater than the time required for data transfer for the simulation to appear real time.

The following table illustrates the time required for various tasks executed on 100 MHz pentium PCs.

Data transfer	**Data size**	**Time (ms)**
Dynamic Data Exchange : Quattro Pro file → LabVIEW VI	Single cell (integer)	≈ 10
Dynamic Data Exchange : Quattro Pro file → LabVIEW VI	Spreadsheet block of 28 X 34 cells (floats)	≈ 70
Dynamic Data Exchange followed by TCP: Quattro Pro file → TCP → VI on remote computer	Single cell (integer)	≈ 60
Dynamic Data Exchange followed by TCP: Quattro Pro file → TCP → VI on remote computer	Spreadsheet block of 28 X 34 cells (floats)	≈ 120
Multiple DDE followed by single TCP: Quattro Pro file → TCP → VI on remote computer	4 spreadsheet block varying from 10 X 1 to 28 X 24 cells (floats)	≈ 150
TCP: large file from one computer to another (including read and write process)	ASCII file 0.67 Mb ≈ 38 000 data point (floats)	≈ 3000 where TCP transfer ≈250 kbytes/sec

Table 1. Administrative time for various tasks

As shown in table 1, the biggest constraint on the communication time between a VI on a remote computer and the blackboard is not so much the amount of data that is transferred but rather the number of communications required for a given task. For instance, the flux technician must extract many blocks of data from the blackboard (dimensions, grid, control parameters, lattice spacing...). Instead of requesting each block separately, the technician sends an array of block names and then is sent back an array of block content. This limits the TCP connection to one. On the server side, a connection with the Quattro Pro file is only established and closed once, reducing furthermore the total required time.

The preliminary results yield that one must allow about 60 ms per simple DDE/TCP communication (eg look-up a single command in blackboard). The communication time is limited by the Ethernet network used at this time. Preliminary results also show that the transfer of large file may hinder the overall data flow. The 0.67 MB file in question in table 1 contains the flux values for a 28 X 34 X 10 grid for four energy groups. However, for the current grid layout, about 10 % of the cells are actually fuel or control cells. Therefore, the flux file that must be transferred to the poison, depletion and thermal hydraulic modules, is significantly smaller. Such file would require less than a second to be transferred from one computer to another. Only if the user requires the complete flux file for display purposes would the transfer last 3 seconds.

Another area that requires further investigation is the handling of many remote computers probing the blackboard at the same time. As a DDE server, Quattro Pro can link with many DDE clients at the same time. This implies that there can be several DDE/TCP servers having different port #s on the computer where the blackboard is located. This approach will only work if the integrity of the data in the

blackboard is not violated. Integrity violation occurs, for example, when a block of data is requested and changed simultaneously or if it is updated and is out of step with the remaining data . This problem is known as the problem of mutual exclusion.

Because the DDE/TCP servers all share the same built- in DDE subVIs and because it can be specified that a subVI is not duplicated when called by more than one VI, only one server really has access to the blackboard at any given time. Then why multiple servers ? This becomes obvious if one realises that a communication with the blackboard occurs in three steps: initialise the connection, transfer the data and terminate the connection. Even though the VI responsible for the data transfer can only be executed by a single server, while a server executes the DDE Request VI, another may execute the DDE Open VI or DDE Close VI. This, in essence, is the first come first served busy waiting mechanism for mutual exclusion. Since the time required for DDE communication is negligible compared to the time required for the TCP connections, the waiting imposed by this scheme should not be a problem.

The last potential problem associated with distributed systems that must be addressed is the sharing of files. Since the flux data, as well as the concentration of various isotopes are stored in an ASCII file and are shared between various modules, one must avoid having a file being both accessed by a reader and writer process. To avoid memory problems, the files are read/written by the C functions themselves. Only when the file is transferred is it ever accessed by a VI. The reader-writer problem as it is known, can easily be avoided if the data is stored in a different file at each time step. Furthermore, since all activities are suspended when a C function is executed by a Call Library Function VI, a file being written can never be accessed by a reading process.

6 Conclusion

The distributed, loosely coupled network of computer programs, organized as a society of cooperating agents using LabVIEW as a platform shows potential for the coupling of simulation codes. The built-in communication functions in LabVIEW as well as the user interface resources contributed greatly to the simplicity of the code. MACSIM's modular structure makes it very flexible and upgradable to real-time data acquisition. Preliminary performance results are promising, although more work is required to improve the logistics and be able to scale up to more than two distinct modules on four or more computers.

Reference

[GAR93] Wm.J. Garland, W.F.S. Poehlman, R.J. Wilson, and A. Bokhari, "Towards a Generic User Support System (GUS), Canadian Nuclear Society Fourth International Conference on Simulation Methods in Nuclear Engineering, Montreal, Canada, June 2-4, 1993.