Simulation of a CIM Workflow System Using Parallel Virtual Machine (PVM)

Chang-Ouk Kim*, Young-Hae Lee**

Abstract

Workflow is an ordered sequence of interdependent component data activities each of which can be executed on an integrated information system by accessing a remote information system. In our previous research [4], we proposed a distributed CIM workflow system which consists of a workflow execution model called DAF-Net and an agent-based integration model of information systems called AIMIS. Given a component data activity, there needs an interaction protocol among agents which allocates the component data activity to a relevant information system. This protocol significantly affects the completion rate of workflow if many information systems exist. The objective of this research is to propose and test two protocols: ARR (Asynchronous Request and Response) protocol and NCL (Negotiation with Case based Learning) protocol. To test the effectiveness of the protocols, we applied the PVM (Parallel Virtual Machine) software to simulate the distributed CIM workflow system. PVM provides a distributed computing environment in which users can run different software processes in different computers while allowing communication among the processes.

1. Introduction

From the viewpoint of information processing, a CIM enterprise can be viewed as a set of distributed and heterogeneous information systems, providing necessary data for manufacturing and production to the staffs. With the rapid development of information network technology, advanced CIM enterprises are evolving toward a more agile framework, which can be adaptable to changing product specifications, manufacturing processes, and market conditions. This demand is driving CIM enterprises to install an integrated information system which is able to (1) automatically coordinate interdependent CIM data activities according to a pre-specified execution sequence for rapidly propagating changes of interest, and (2) connect a variety of distributed and heterogeneous information systems by demand. In this research, the CIM data activity implies not only database transaction but also file copy, file transfer, e-mail, and an

* Korea University

** Department of Industrial Engineering, Hanyang University

execution of application software.

For the past fifteen years, the areas of information processing have been extensively researched in the industrial engineering and computer science communities [1][3][5][6]. However, there are two limitations of the previous research. One is that the research has focused on the (schema) integration of databases or the relatively simple coordination of CIM data activities using rule-base (active database) systems. The other is that few models proposed by the research has been simulated to test the execution logic of the models or to decide some design features of the models.

To resolve the first limitation, Data Activity Flow Net (DAF-Net) model and Agent-based Integration Model for Information Systems (AIMIS) model have been developed in out previous research [4]. DAF-Net is a CIM workflow which is triggered in response to events of interest. It can be defined as a set of component data activities with an execution sequence among them. According to the execution sequence, each component data activity in a given DAF-Net is delegated to an information system for its execution. AIMIS is a multi-agent based integration model which can integrate new heterogeneous information systems by demand. It is also designed to support the concurrent implementation of DAF-Net.

The objective of this paper is to investigate the Parallel Virtual Machine (PVM) software for the simulation of DAF-Net and AIMIS. PVM is a distributed computing software [2]. It provides a distributed computing environment in which users can run different software processes in different computers while allowing communication among the processes. There are three advantages of PVM compared to traditional simulation tools such as SLAM: (1) PVM can physically simulate (emulate) the interactions (message passing) of agents in a distributed multi-agent system, (2) PVM can also simulate a multi-tasking logic of AIMIS, which will be discussed in the next section, and (3) the execution logic of agents can be simulated using PVM coupled with the C language. In this paper, using PVM, two agent interaction protocols are compared in terms of the number of messages passed.

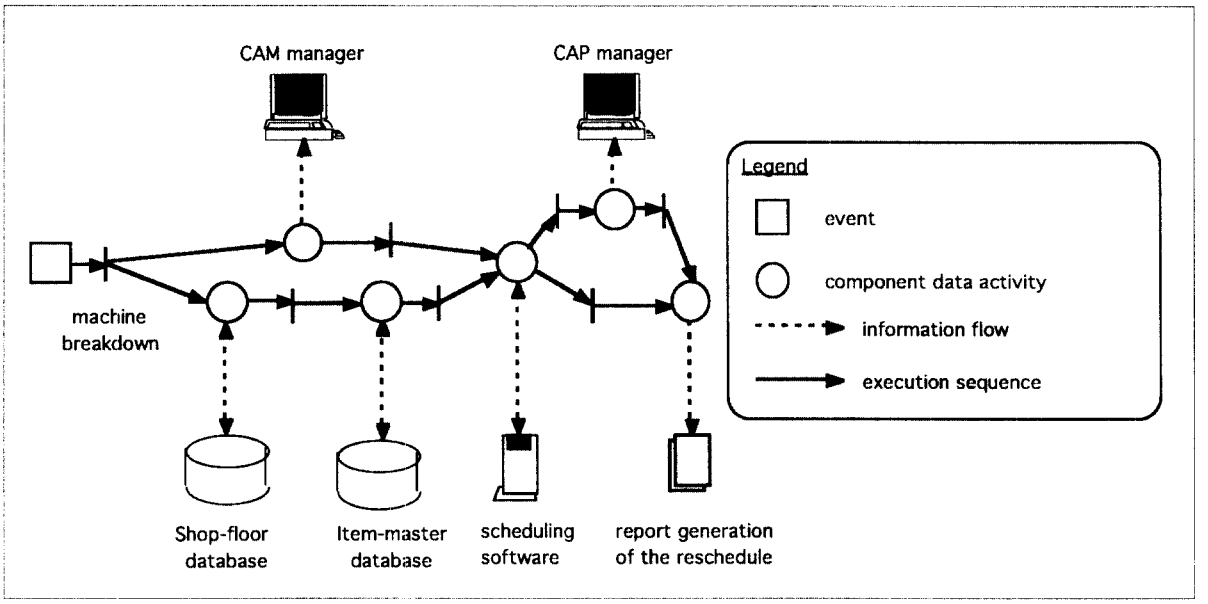
This paper consists of five sections and is organized as follows. In Section 2, DAF-Net and AIMIS are briefly illustrated. In Section 3, PVM is introduced, and its communication functions for the simulation of DAF-Net and AIMIS are presented. In Section 4, a simulation model and its result are presented. Finally, in Section 5, the conclusion of this research and the future research work are discussed.

2. DAF-Net and AIMIS

2.1 DAF-Net

DAF-Net is defined for each event e_i $(i=1,2,\cdots,L)$and constructed to achieve a pre-defined goal which compensates the event. Therefore, the relation between event and DAF-Net is one-to-one mapping. Informally, DAF-Net D_i $(i=1,2,\cdots,L)$ is a graphical execution model of interdepen dent component data activities whose execution should be coordinated responding to event e_i $(i=1,2,\cdots,L)$. Formally, DAF-Net D_i $(i=1,2,\cdots,L)$ defined as a five-tuple $D_i=(C_i, TR_i, IN_i, OUT_i, MR_i)$. C_i is a set of component data activities $\{c_{i1},c_{i2},\cdots,c_{i|C_i|}\}$ where $|C_i|$ is the size of C_i. Each component data activity should be executed by delegating it to a relevant information system which stores data for the component data activity. TR_i is a set of transitions $\{t_{i1},t_{i2},\cdots,t_{i|TR_i|}\}$ where $|TR_i|$ is the size of TR_i. IN_i is an input function that defines directed arcs from C_i to TR_i. OUT_i is an output function that defines directed arcs from TR_i to C_i, and MR_i is the marking of D_i(execution state of D_i). TR_i, IN_i, and OUT_i are used for explicitly representing the concurrency and synchronization of component data activities.

Structurally, DAF-Net follows the Petri net model. However, unlike traditional applications of Petri net such as performance evaluation and the control of manufacturing cell [8][9], DAF-Net coordinates a set of interdependent CIM data activities in response to an event in real-time mode. In other words, in the traditional applications of Petri net, each place denotes a certain state of a given system. But, in DAF-Net, each place represents a component data activity which should be delegated to an information system for its



〈Figure 1〉 Rescheduling DAF-Net : An example of CIM DAF-Nets.

execution.

〈Figure 1〉 shows a rescheduling DAF-Net to a machine breakdown event. Each circle denotes a component data activity and a rectangle means an event. Also, execution sequence is represented by a solid line with an arrow, and information flow generated by each component data activity is represented by a dotted line with an arrow. The rescheduling DAF-Net consists of six component data activities - retrieval transaction of resource status data in shop-floor database, retrieval transaction of due-date of parts from item-master database, generation of reschedule, notification to CAP and CAM managers, and report generation of the reschedule. By the collaboration of the six component data activities, rescheduling to the machine breakdown (goal) can be completed.

2.2 AIMIS

For realizing the "integration by demand" concept, AIMIS follows a multi-agent architecture and a open system model. It is composed of two types of agents: global coordination agents (GCAs) and local coordination agent (LCAs). 〈Figure 2〉 depicts the architecture of AIMIS.

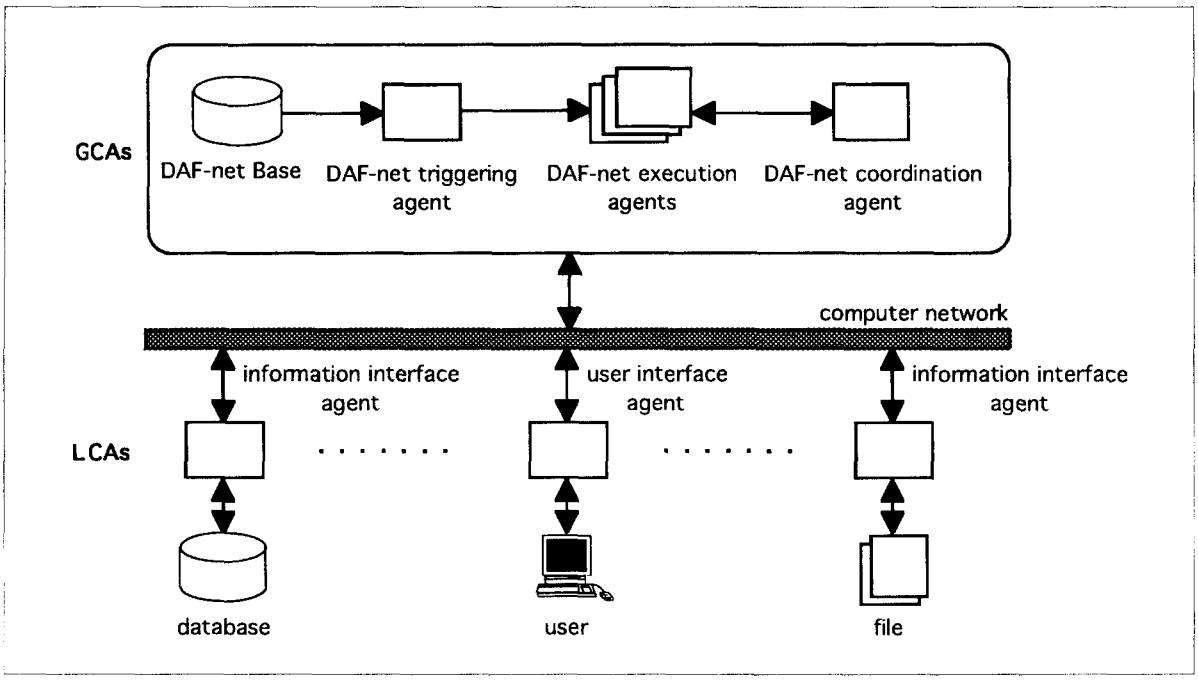
GCAs reside in a central computer. GCAs are classified into three types of agents based on the function of each agent: a DAF-Net triggering agent G_T, a set $G_E=\{g_1^E, g_2^E, \cdots g_n^E\}$ of DAF-Net execution agents where n is the number of DAF-Nets being executed, and a DAF-Net coordination agent G_C. Each type of the GCAs performs the following function:

a. DAF-Net triggering agent (G_T)

G_T receives events from LCAs and fetches relevant DAF-Nets (matching DAF-Net) from the DAF-Net library in response to the events. It also spawn a DAF-Net execution agent g_i^E to execute a DAF-Net D_i

b. DAF-Net execution agent (G_E)

g_i^E ($i = 1, 2, \cdots, n$) controls the execution sequence of component data activities of a DAF-Net to enforce their execution sequence. According to the execution sequence, g_i^E sends component data activities to appropriate LCAs and receives the execution results from the LCAs.



〈Figure 2〉 Architecture of AIMIS.

c. DAF-Net coordination agent (G_c)

G_c controls the execution sequence of multiple DAF-Nets. In other words, G_c performs the concurrency control of DAF-Net.

One particular feature of AIMIS is that the number of the DAF-Net execution agents $G_E=\{,g_1^E,g_2^E,\cdots g_n^E\}$ is not fixed but varies with the number of DAF-Nets being executed. That is, the G_E do not permanently reside in AIMIS; rather each of them, say g_i^E, is created by the G_T to execute DAF-Net D_i and terminates after the each execution completion of D_i. In this research, this type of agent is called an *ephemeral agent*. The dynamic agent environment using the concept of ephemeral agent can improve the performance of AIMIS in terms of the completion rate of DAF-Net. To implement the concept of the ephemeral agent, a multi-tasking method is applied.

Each LCA is placed on an information systems and connected to both the information system and GCAs. Thus, each information system is augmented by an LCA which provides a coordination interface.

Besides the functions of the agents, an agent interaction protocol between GCAs and LCAs is required for processing DAF-Nets. Given a component data activity, the function of the agent interaction protocol is to search an appropriate information system in terms of data availability and processing efficiency for the execution of the component data activity. In this research, such an interaction protocol is called *the allocation protocol of component data activity*. Two protocols are proposed: ARR (Asynchronous Request and Reply) and NCL (Negotiation with Cased based Learning) protocols. Since the coordination of interdependent CIM data activities on distributed information systems requires a lot of interactions among the agents, the performance of AIMIS is significantly affected by the allocation protocol of component data activity. The effectiveness of the two protocols is tested by PVM. The ARR protocol and the NCL protocol are procedurally explained as follows:

Asynchronous request and response (ARR) protocol

The ARR protocol is the most efficient communication protocol in a client-server model because of the small number of messages passed between g_i^E and LCAs for completing DAF-Net D_i. It can be specified in the following way.

Step 1. Using the address of the receiver (LCA) field of component data activity specification, g_i^E sends the request to the LCA and waits for a response (acknowledgment) from the LCA.

Step 2. If the LCA scans its request queue and finds the arrival of the request, then it sends an acknowledgment signal to the g_i^E.

The main advantage of the ARR protocol is that only one round-trip message is required to find the location of the LCA. The major drawbacks of the protocol are that (1) the address of receiver must be specified for every component data activity and (2) there is a chance of infinite waiting of g_i^E to receive acknowledgment from the LCA which is currently broken down.

Negotiation with case-based learning (NCL) protocol

The NCL protocol relaxes the assumption of a priori knowledge about the addresses of the LCAs. Instead, this protocol adopts a negotiation mechanism between g_i^E and the LCAs to find the most appropriate LCAs which can execute the component data activities efficiently.

One remarkable advantage of the NCL protocol is its learning ability. After each negotiation, g_i^E accumulates the processing capability of each LCA in the LCA capability table. That is, each negotiation makes the g_i^E learn which LCA stores which type of data. The LCA capability table is a triple (*LCA identifier, LCA address, list of data objects*) where the *LCA identifier* is a unique identifier of LCA, *LCA address* is the physical address of the LCA on the computer network, and *list of data objects* is the list of data objects that are accessible by the LCA.

As the interaction between g_i^E and the LCAs evolves, g_i^E gains accurate knowledge of the LCAs' capability and can send limited broadcasting messages, which will reduce the communication traffic between g_i^E and LCAs. The main disadvantage of the NCL protocol is that it generates too many communication messages when data stored in each information system dynamically change. The protocol can be explained procedurally as follows:

Step 1. g_i^E searches the LCA capability table to find LCAs which are able to execute a given component data activity. The selected LCAs are called candidate LCAs. If only one candidate LCA exists, go to Step 5. If no candidate LCA exists, then go to Step 2-b.

Step 2-a. g_i^E broadcasts a component data activity to all candidate LCAs and waits for their responses. Go to Step 3.

Step 2-b. g_i^E broadcasts a component data activity to all LCAs and waits for their responses.

Step 3. Each LCA makes a bid. It consists of (1) the possibility of the execution of the component data activity based on the availability of data required for the execution and (2) the number of component data activities waiting in its request queue.

Step 4. LCAs send their bids to g_i^E .

Step 5. g_i^E selects a winning LCA which not only can execute the component data activity but also has the smallest size of the request queue. In addition, g_i^E updates the processing capability of the winning LCA.

Step 6. After the negotiation procedure, g_i^E sends an awarding message to the winning LCA.

3. Parallel Virtual Machine (PVM)

In this section, PVM software is reviewed. The overall objective of the PVM software is to enable a collection of heterogeneous computers to be used cooperatively for concurrent or parallel computation. The PVM software provides a unified framework within which parallel programs can be developed in an efficient and straightforward manner using existing hardware. PVM enables a collection of heterogeneous computer systems to be viewed as a single

〈Table 1〉 Communication routines of the PVM used in the experiments.

Communication routine	Description of the routine
pvm_spawn ()	Creates a computational module on computer
pvm_exit ()	Tells that the process is leaving PVM
pvm_initsend ()	Clears default send buffer and specifies message encoding
pvm_send ()	Sends the data in the active message buffer
pvm_pk ()	Packs the active message buffer with arrays of prescribed data type
pvm_upk ()	Unpacks an array of the given data type from the active receive buffer
pvm_recv ()	Receives a message and blocks the process until a message with label msgtag has arrived
pvm_nrecv ()	Checks for mon-blocking message with label msgtag

parallel virtual machine, and transparently handles all message routing, data conversion, and task scheduling across a network of incompatible computer architecture.

Since PVM accommodates a wide variety of application program structures, the user can write his application as a collection of cooperating tasks. Tasks access PVM resources (computers) through a library of standard interface routines. These routines allow the initiation and termination of tasks across the network. The PVM message passing primitives are oriented towards heterogeneous operation, involving strongly typed constructs for buffering and transmission. Communication constructs include those for sending and receiving data structures as well as high level primitives such as broadcasting, barrier synchronization.

Due to the simple but complete programming interface, PVM is employed in this research in order to (1) simulate the interactions of agents in AIMIS to process DAF-Nets, (2) simulate the multi-tasking logic of DAF-Net execution agents (ephemeral agents), and (3) simulate the allocation protocol of component data activity. Unlike to other research where PVM assists to solve a single large scientific problem, this research applies PVM to simulate interactive protocols of AIMIS in terms of number of messages passed. This unique fact extends the PVM application area and can be possibly used to measure performance of the distributed agent system. 〈Table 1〉 shows the interaction routines of PVM used in this research

To simulate the ARR and NCL protocols on PVM environment, it is necessary to support communication function among agent. This communication is realized by address keeping mechanism in PVM. That is, each processor which simulates an agent(GCA or LCA) stores the logical addresses of all other processes. This enables an agent to communicate with other agents for either the negotiation of the NCL protocol or the asynchronous sending/receiving of the ARR protocol. The physical communication is undertaken by PVM. The following procedure illustrates an interaction between any two agents in PVM. 〈Figure 3〉 graphically illustrates the above interaction mechanism.

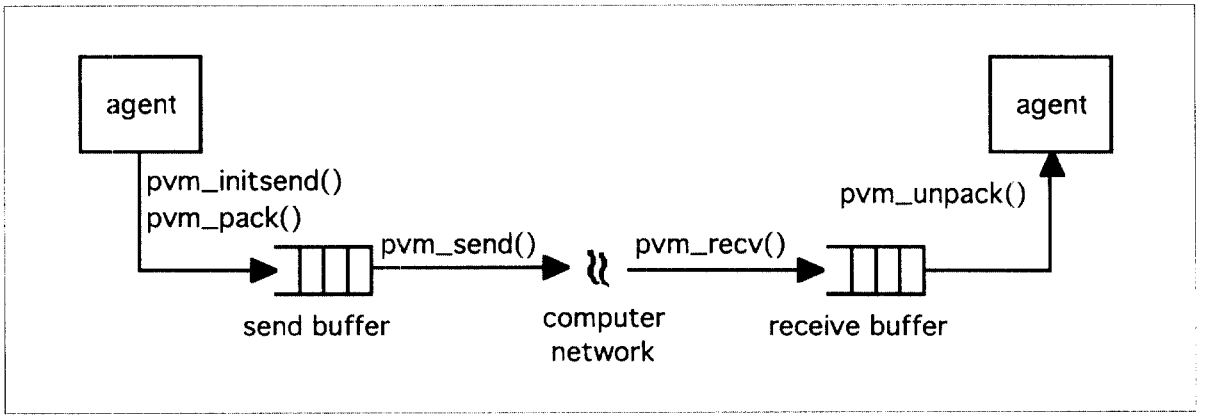
Task of message-sending-agent

1. Initializes its send buffer by a call to pvm_initsend() routine.
2. Packs message into this buffer using pvm_pk() routine.
3. Sends the packed message to message-receiving-agent by calling pvm_send() routine.

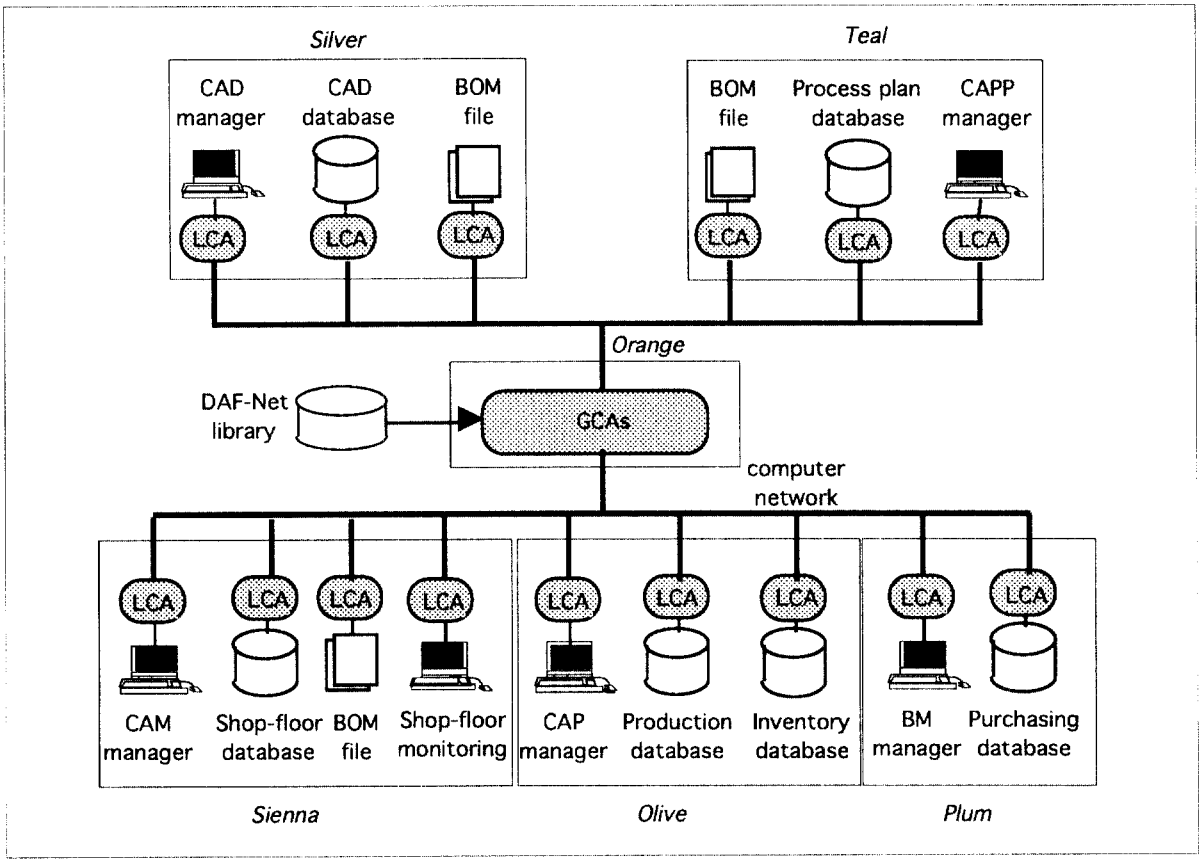
Task of message-receiving-agent

1. Receives the packed message by calling either pvm_recv () (blocking receive routine) or pvm_nrecv() (non-blocking receive routine).
2. Unpacks the message from its receiver buffer using pvm_pk() routine.

The following procedure shows the multi-tasking logic of the DAF-Net execution agents.



〈Figure 3〉 PVM-based interaction between two agents in AIMIS.



〈Figure 4〉 Simulation model of MRPII scenario and PVM configuration.

Task of DAF-Net triggering agent

1. When an event arrives, retrieve a matching DAF-Net from DAF-Net library.
2. Create a DAF-Net execution agent using pvm_spawn() routine.
3. Initializes its send buffer by a call to pvm_initsend() routine.
4. Packs the matching DAF-Net into this buffer using pvm_pk() routine.
5. Sends the packed matching DAF-Net to the newly created DAF-Net execution agent by calling pvm_send() routine.

Task of newly created DAF-Net execution agent

1. Receives the packed matching DAF-Net by calling pvm_recv().
2. Unpacks the matching DAF-Net from its receiver buffer using pvm_pk() routine.
3. Process the matching DAF-Net using the agent interaction mechanism in 〈Figure 3〉.
4. After the processing completion of the matching DAF-Net, terminate itself by calling pvm_exit() routine.

4. Simulation of AIMIS

Because of the distributed nature of AIMIS, Engineering Computer Network (ECN) in Purdue University is used to run PVM. A virtual MRPII scenario is selected as a typical CIM environment. To implement a realistic MRPII system, it is essential to coordinate and integrate several component heterogeneous information systems from a CAD file to an accounting database. This requirement led to the simulation of an MRPII system as the scenario of the experiments.

The PVM software and the C language are employed as tools for constructing the distributed simulation program. Since PVM is developed for facilitating distributed computing on a computer network, it would be an appropriate tool for simulating DAF-Net and the interactive behavior of the distributed agents.

The simulation scenario of MRPII consists of five departments as shown in 〈Figure 4〉. Two heterogeneous information systems and a manager are assumed to be associated with each department, except the business management department, where only one information system is assumed. For interoperability of the information systems, an integrated information system which follows the AIMIS approach is constructed. All LCAs are connected to GCAs, and each LCA is linked to each information system or manager. For this simulation, eight event types are chosen as follows:

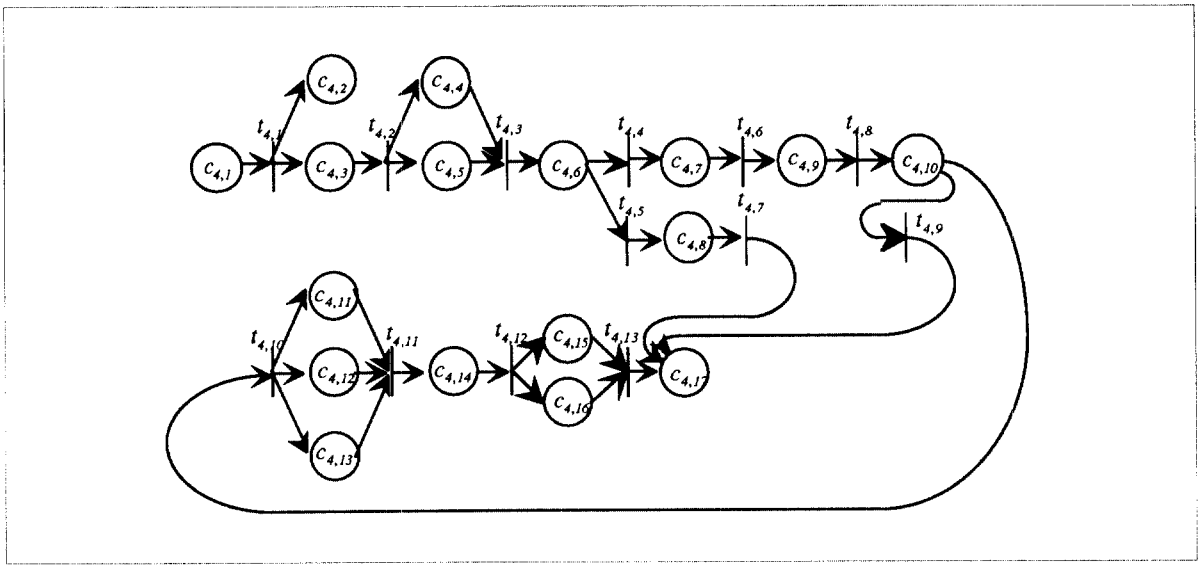
a. Inventory update;
b. Change of CAD and BOM data;
c. Change of cost and lot-size data;
d. Change of part lead-time;
e. Shop-floor order completion;
f. Machine breakdown;
g. Due-date change of scheduled receipt;
h. Daily production progress report to production manager.

Corresponding to each event type, a DAF-Net is constructed. 〈Figure 5〉 shows one of the DAF-Nets.

To use the PVM software for the MRPII simulation, 33 processes are run simultaneously on six Unix-based workstations which are interconnected by Ethernet, all part of the ECN. Each processor is an independent program. It emulates the behavior of either an information system, user (manager), or agent (GCA or LCA) of AIMIS. Some processes are loaded and executed on the same workstation (shown as a box) due to the limited number of workstations available. 〈Table 2〉 depicts the configuration of the processes with PVM.

In general, successful applications of a multi-agent model depend on the number of messages passed among distributed agents. A high completion rate of distributed tasks with a limited or moderate amount of message passing would be the main goal of the interaction protocol. This goal is also applied to AIMIS models. The design feature of AIMIS that significantly affects the number of messages passed is the allocation protocol of component data activity. Therefore, it is necessary to compare the ARR protocol and the NCL protocol.

To compare the above two protocols in the MRPII



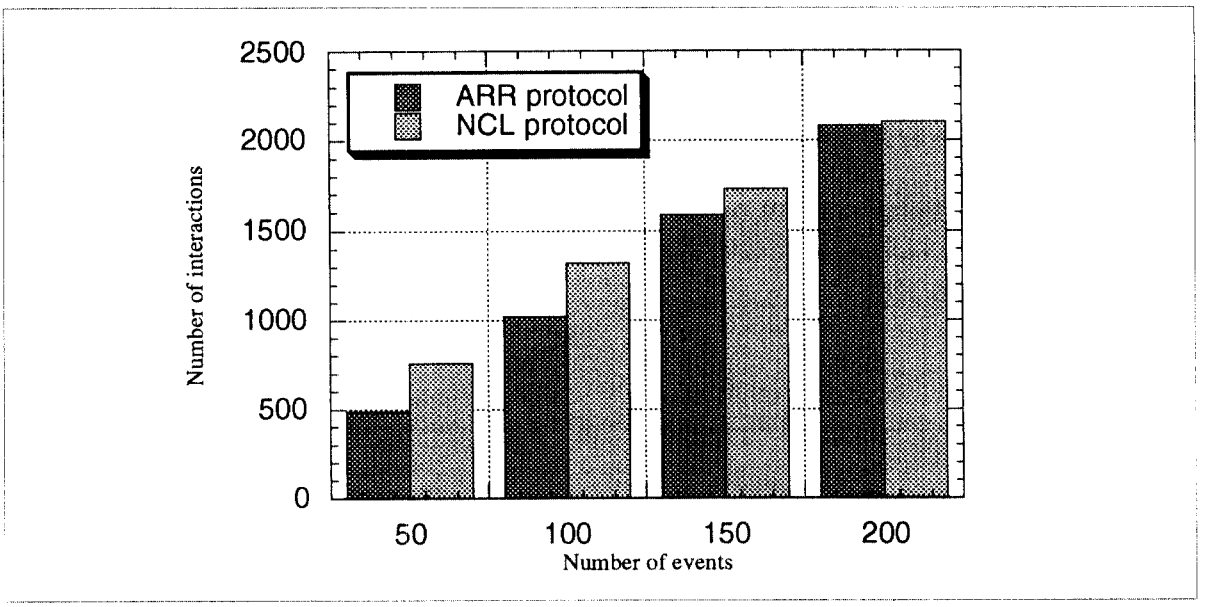
Activity	Type	Information system	Description
$C_{4,1}$	Null	Null	Starting dummy node
$C_{4,2}$	Alarm message	E-mail	Notift CAM manager of "lead-time change"
$C_{4,3}$	DB transaction	Production DB	Retrieve part data whose lead-time is changed
$C_{4,4}$	DB transaction	Shop floor DB	Retrieve resource status data
$C_{4,5}$	DB transaction	Process plan DB	Retrieve the routing data of the part
$C_{4,6}$	Condition evaluation		Calculate new expected lead-time and if the expected lead-time is greater than changed lead-time take $t_{4,5}$ else $t_{4,4}$
$C_{4,7}$	Alarm message	E-mail	Notify CAP manager of "lead-time change"
$C_{4,8}$	Alarm message	E-mail	Send warning message to CAP manager
$C_{4,9}$	Waiting	Null	Waits until CAP manager sends a response
$C_{4,10}$	Condition evaluation		If response is "continue", then take $t_{4,10}$ else $t_{4,9}$
$C_{4,11}$	DB transaction	Production DB	Retrieve MPS data
$C_{4,12}$	DB transaction	Inventory DB	Retrieve inventory data
$C_{4,13}$	DB transaction	Production BD	Retrieve item master data
$C_{4,14}$	Application software	MRP software	Execute MRP software
$C_{4,15}$	Application software	MRP software	Execute MRP report generation program
$C_{4,16}$	Alarm message	E-mail	Notify CAP manager of "MRP update"
$C_{4,17}$	Null	Null	Ending dummy node

〈Figure 5〉 DAF-Net for the change of part lead-time.

〈Table 2〉 Configuration of the processes with PVM.

Simulation module	Process number	Computer name
DAF-Net triggering agent (GCA)	1	Orange
DAF-Net execution agents (GCAs)	2	Orange
DAF-Net coordination agent (GCA)	3	Orange
LCA for CAD DB	4	Silver
LCA for BOM file in CAD dept.	5	Silver
LCA for CAD manager	6	Silver
CAD DB	7	Silver
BOM file in CAD dept.	8	Silver
CAD manager	9	Silver
LCA for process plan DB	10	Teal
LCA for BOM file in CAPP dept.	11	Teal
LCA for CAPP manager	12	Teal
Process plan DB	13	Teal
BOM file in CAPP dept.	14	Teal
CAPP manager	15	Teal
LCA for shop-floor DB	16	Sienna
LCA for BOM file in CAM dept.	17	Sienna
LCA for CAM manager	18	Sienna
LCA for shop-floor monitoring	19	Sienna
Shop-floor DB	20	Sienna
BOM file in CAM dept.	21	Sienna
CAM manager	22	Sienna
Shop-floor monitoring	23	Sienna
LCA for production DB	24	Olive
LCA for inventory DB	25	Olive
LCA for CAP manager	26	Olive
Production DB	27	Olive
Inventory DB	28	Olive
CAP manager	29	Olive
LCA for purchasing DB	30	Plum
LCA for BM manager	31	Plum
Purchasing DB	32	Plum
BM manager	33	Plum

scenario, the number of messages passed is measured as the number of events varies from 50, 100 through 150 to 200. 〈Figure 6〉 shows the simulation result. From the result, it is observed that the ARR protocol is better than the NCL protocol in terms of the number of messages passed. However, it is also observed that the ARR protocol is superior to the NCL protocol until the number of events reaches 100, but thereafter, there is no significant difference between the protocols. This result can be explained by relying on the nature of NCL protocol. Since the NCL protocol has the ability to learn the location of necessary data, and this capability is enhanced as the number of events increases,



〈Figure 6〉 Simulation result for the comparison between the ARR protocol and the NCL protocol.

the NCL protocol reaches a point in time at which the NCL protocol has as much knowledge on the data location as the ARR protocol. From the point in time, the learning capability of the NCL protocol led the NCL protocol to reduce the number of negotiation. Consequently, it can be inferred that the intelligent protocol, capable of learning knowledge from past experience, is valuable when the integrated system is operated for a longer time.

5. Conclusion

Throughout this research, PVM has been investigated for the simulation of distributed CIM workflow system. From the simulation result, it is verified that PVM is a highly effective simulation tool for distributed systems. For future research, it is necessary to develop a PVM-based distributed simulator which can measure time-related performance measures, such as the mean completion time of DAF-Net.

References

[1] Etzion, O., Dori, D., and Nof, S.Y., "Active Coordination of a CIM Multi-Database System," *Research Memorandum* No. 93-17, School of Industrial Engineering, Purdue University, 1993.

[2] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V., *PVM: Parallel Virtual Machine: A User's Guide and Tutorial for Networked Parallel Computing*(MIT Press), 1994.

[3] Hsu, C., Gerhardt, L., Spooner, D., and Rubenstein, A., "Adaptive Integrated Manufacturing Enterprises: Information Technology for the Next Decade," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 24, 828-837, 1994.

[4] Kim, C.O., *DAF-Net and Multi-Agent Based Integration Approach for Heterogeneous CIM Information Systems*, Unpublished Ph.D. Dissertation, Purdue University, 1996.

[5] Libes, D. and Barkmeyer, ED., "The Integrated Manufacturing Data Administration System(IMDAS)- An Overview," *Int. J. of Computer Integrated Manufacturing*, vol. 1, 44-49, 1988.

[6] Urban, S.D., Shah, J.J., Rogers, M., Jeon, D.K., Ravi, P., and Bliznakov, P., "A Heterogeneous, Active Database Architecture for Engineering Data Management," *Int. J. Computer Integrated Manufacturing*, vol. 7, 276-293, 1994.

[7] Vernadat, F., Artificial Intelligence in CIM Databases, in *Artificial Intelligence : Implications for CIM*, (Kusiak, A. ed.), Springer-Verlag, 1988.

[8] Viswanadham, N. and Narahari, Y., *Performance Modeling of Automated Manufacturing Systems*(Prentice Hall), 1992.

[9] Zhou, M. and Dicesare, F., *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*(Kluwer Academic Publisher), 1993.

● 저자소개 ●

김창욱

1988년 고려대학교 공과대학 산업공학과 졸업
1990년 고려대학교 대학원 산업공학과 졸업 (석사)
1996년 미국 Purdue대학교 대학원 산업공학과 졸업 (박사)
현 재 고려대학교 정보통신기술연구소 선임연구원

이영해

1977년 고려대학교 산업공학 학사
1983년 미국 Univ. of Illinois, 산업공학 석사
1986년 미국 Univ. of Illinois, 산업공학 박사
1990년 일본 오사카대학 전자제어기계공학과 객원교수
현 재 한국시뮬레이션학회 부회장, 한양대학교 산업공학과 교수
관심분야 : Simulation in Manufacturing, Simulation Output Analysis, Simulation Optimization.