An Expert System for Performance Management

Manish Parashar, Salim Hariri and Kamal Jabbour

Abstract

The rapid growth in size and complexity of distributed systems and the use of heterogeneous components require effective tools to manage their resources. Expert system technology possesses the potential to manage such large and complex systems based on heuristic knowledge in near real time. This paper presents a general approach to designing an expert system for the management of performance of a distributed function which is built based on the services provided by several application servers distributed across the network. Furthermore, the expert system re-allocates the application servers so that all the clients (who are running the same distributed function from different sites) will experience similar function response times. We also present an implementation of this system using a commercial expert system shell and inference engine. This presented expert system design has three components: (1) resource monitor, (2) performance optimizer, and (3) the Management Information Base (MIB). We demonstrate through an example how this expert system can be used to optimize the distribution of application servers of a client-server based distributed system with three ethernet-based networks interconnected by an FDDI backbone network.

1 Introduction

Distributed computing systems can be described as networked systems of applications, high-performance computers, workstations, and special purpose servers. The underlying network of a distributed system ranges from a high-speed bus that connects a cluster of computers to a wide area network. The current advances in local area networks (LAN's) have had a significant impact on the wide spread of distributed systems. As the size and the complexity of a distributed system network increases, the management of this network and its resources is becoming a challenging problem. The network management tasks include a set of operational and administrative functions required to bring up a network, keep it operational and fine tune its operation. The management framework being developed by the International Standardization Organization (ISO) allows the management of a multi-vendor network components in a unified manner. The ISO management model consists of the following areas [1, 5, 6, 7]: 1) Performance management; 2) Configuration management; 3) Fault management; 4) Accounting management; and 5) Security management.

The focus of the paper is on performance management which can be defined as the set of tools and functions needed to guarantee that the network meets its performance goal. In particular, we present an approach that uses expert system technology to minimize the execution time of a distributed function, and at the same time reduces the variance of its response time as perceived by users at sites across the network. The distributed computing model adopted in this paper is based on the client-server paradigm. In the client-server model, the user function (client function) is built based on the services provided by several Application Servers (AS). For example, an engineering client program needs the services provided by several file servers and compute servers; or a command, control, and communication client function will need the services provided by compute, database, image, and graphics servers [2].

In this paper, we propose a generalized architecture for an expert system to manage the performance of distributed applications. We apply the proposed architecture to construct an expert system for allocating the resources needed by a distributed function. The resources are allocated such that the function response time is minimized and the system is well balanced with respect to all client sites. The expert system is implemented using the CLIPS (C Language Implementation Production System) inference engine and contains the following components: Resource Monitor, Performance Optimizer and Management Information Base (MIB). The MIB stores all the information relevant to the managed objects in the system. The monitor accesses this data from the MIB using standard access protocols and uses it to estimate the current performance metrics of the client functions to be managed. The monitor invokes the performance optimizer if the response time of the managed function has deteriorated below a predetermined threshold. The performance optimizer then attempts to obtain an allocation of resources that has an acceptable function response time. If it fails to obtain such an allocation the performance optimizer reports the status of the system to the user and seeks the help of the system administrator.

The organization of the rest of the paper is as follows. In Section 2, we present an approach to build an expert system for performance management. Section 2.2 describes an implementation of the system using the CLIPS inference engine. Section 2.3 presents a numerical example to demonstrate how this expert system can be used to minimize the response time of a function that requires the services of application servers distributed across three ethernet networks interconnected by an FDDI backbone network. In Section 3, we provide a summary and concluding remarks.

2 An Expert System for Performance Management

There is a rapidly growing interest in Artificial Intelligence and Expert System technology and their application to resource allocation and network management. This interest is evidenced by some Expert Systems that have proved helpful for fault-diagnosis, system testing and monitoring, systems engineering, intelligent user interfaces, and network management. Expert systems are different from conventional software systems because they separate system knowledge or rules from program control and data [10]. They result from an interactive development process with customer participation. Expert systems can make a “best guess” at why a problem occurred, cope with incomplete or inaccurate data, and sometimes explain how they arrived at a conclusion. Some operational systems even demonstrate the ability to anticipate trouble and learn from experience.

The AT&T Automated Cable Expertise (ACE) system is used to identify cable routes needing rehabilitation and preventive maintenance in order to anticipate and thereby avoid customer trouble reports [10]. ACE was a pioneering application in that it demonstrated that knowledge based systems could be successfully employed in telecommunications. EXNETS is another example on using an expert system to detect and diagnose network faults and invoke proper recovery procedures [11]. Also, reference [12] presents the ELAND expert system that can be used to design local area networks. We believe that expert systems have the potential to successfully manage the performance of functions in a distributed computing environment. In what follows, we describe an approach for using expert system techniques to manage resource allocations of a heterogeneous distributed environment such that the client function response time is minimized with respect to all client sites.

2.1 Design Approach

A typical distributed environment consists of a set of possibly heterogeneous computing platforms which are interconnected through a network and are capable of cooperating with each other in handling applications. Distributed applications running on such an environment are typically divided into client programs and application servers [2]. Client programs are consumers of resources; application servers provide the resources and can be distributed on different sites across the network.

In the design approach presented in this paper, the expert system is designed to operate in the background in cooperation with the distributed operating system. It consists of three main subsystems: (1) The Management Information Base, (2) The Resource Monitor and (3) The Performance Optimizer (see Figure 1.) The Resource Monitor obtains system information by accessing the Management Information Base (MIB) and then computes the Function Response Time (FRT) at each site. If the FRT at any site increases above a predefined acceptable maximum, the expert system activates the Performance Optimizer subsystem. The performance optimizer then, attempts to find an optimal allocation of resources so as to minimize the response time while maintaining a balance across the client sites. The individual subsystems are described below:

2.1.1 Management Information Base (MIB)

The MIB presents a well-defined abstract image of the managed objects in each of the seven layers of the OSI reference model. The actual objects are contained in a real database in each node. The objects are examined or changed by manipulating the virtual objects in the MIB. A suite of standard application-layer protocols is used to access the MIB objects. It is important to note that the MIB must span all seven OSI layers and be able to access and change management information in each of the seven layers. The MIB thus provides the Resource Monitor with the required system pa-

In the above equation, $\rho \equiv \lambda\ m$ where λ is the total average traffic in frames/sec; m is the average packet length in units of time and $\overline{m^2}$ is the second moment of the frame length distribution. The function $F_p(\lambda)$ is the Laplace transform of the frame length distribution $f(t)$ obtained as follows:

$$F_p(\lambda) = \int_0^\infty f(t)\ e^{-\lambda t} dt$$

a is defined as the ratio τ/m, τ being the end-to-end propagation delay.

Token Ring LAN

$$T_{pkt} \quad = \quad \frac{L(1-\rho/N)}{2(1-\rho)} + \frac{\lambda}{2}\frac{\overline{m^2}}{1-\rho} + m + \frac{L}{2} \qquad (5)$$

Here N is the number of stations along the ring and L is the ring latency or total walk time given by

$$L \equiv \tau + Nb/C$$

C being the transmission capacity.

FDDI LAN

$$T_{pkt} = \frac{\frac{\rho N}{2(1-\rho)}[\lambda\ r^2 + r(1-\rho_i)] + \frac{r\rho^2}{k(1-\rho} - \frac{\rho(1-\rho_i)}{2\lambda k} g^{(2)}}{\rho[1-\frac{g}{k}]} \qquad (6)$$

where ρ is the total offered traffic for N stations such that $\rho \equiv N\rho_i$. Packet arrivals at each station are assumed to Poisson with mean value λ. r is the propagation delay between nearest stations and g is the average number of packets served at each station in one token rotation, while k is the maximum number of packets served at each station during one token rotation and is given by

$$k = \lceil \frac{T\_Opr - Nr}{t * N} \rceil$$

Here T_Opr is the operative target token rotation time and t is the packet transmission time (packet lengths being constant).

After computing the FRT at each client site, the Resource Monitor compares the computed FRT with the pre-defined maximum acceptable FRT (Max_Acc_FRT). If the FRT at any site is found to be greater than Max_Acc_FRT, the Resource Monitor triggers the Performance Optimizer which is described below. The Resource Monitor is invoked at regular intervals to monitor the system performance and the above procedure is repeated.

2.1.3 Performance Optimizer

The Performance Optimizer is responsible for optimizing the resource allocation and system load whenever an unacceptable increase in FRT is detected. The optimizing strategy followed attempts to distribute the application servers based on their characteristics and the current network parameters so as to minimize the following two parameters; (1) the average FRT across the system and (2) the variance of the FRT across various client sites. i.e. so that clients at every site experience similar acceptable function response times. When invoked, the performance optimizer sorts the clients sites in ascending order of their FRT's and identifies the sites with the worst performance (MAX_FRT) and the best performance (MIN_FRT). The current average FRT (Avg_FRT) is also computed as follows:

$$Avg\_FRT = \frac{1}{\#Sites} \sum_{i=1}^{\#Sites} FRT(S_i) \qquad (7)$$

In addition, the performance optimizer maintains two variables which store the current state of the system. These are $CurAvg\_FRT$ which stores the current average FRT and Cur_Config which records the current network configuration. Optimization is achieved using an iterative process wherein, in each iteration the optimizer attempts to improve the the average FRT by migrating a server from the site with MIN_FRT to the site with MAX_FRT. This migration is justified by the observation that the major contributor to the FRT in a network based distributed system is the increased network latency. MIN_FRT implies that most of the required servers are available either locally or the number of communication links traversed is small. Similarly, MAX_FRT implies large communication latency. Increased utilization of a server could also result in an increase in the FRT. However this increase would be reflected in the FRT for every site and hence the site with the maximum FRT will again be the site which experiences the maximum network latency. The resource chosen for migration at the site with MIN_FRT is the smallest resource with respect to the total size of the communication associated with it, in the execution of the function F. If the migration results in an improved FRT, it is accepted and the values of $CurAvg\_FRT$ and Cur_Config are updated, else the migration is rejected and alternate migrations are explored. The steps of the optimization algorithm is stated below:

1. Sort FRT's in an ascending order; Identify MIN_FRT, MAX_FRT; Compute Avg_FRT.

2. Update system variables $CurAvg\_FRT$ and Cur_Config.
3. Sort the resources at the site with MIN_FRT in an ascending order by size; Migrate the smallest migratable resource from the site with the MIN_FRT to the site with the MAX_FRT.
4. Recompute FRT at each site. This can be accomplished analytically or can be approximated.
5. Calculate new average FRT ($NewAvg\_FRT$) and compare with the stored $CurAvg\_FRT$.
6. **If** ($NewAvg\_FRT < CurAvg\_FRT$)
    - Accept the current configuration and update Cur_Config.
    - Assign $NewAvg\_FRT$ to $CurAvg\_FRT$.

    Else

    - Discard migration.
7. Repeat above steps till some convergence criterion is met. This could means selecting the first configuration for which FRT $\leq$ Max_Acc_FRT for each site or finding the configuration with the lowest FRT or any other predefined criterion.
8. If no configuration is found with FRT $\leq$ Max_Acc_FRT, the Performance Optimizer reports the status of the system and seeks human intervention.

2.2 A CLIPS based implementation

In this section we present an implementation outline of the expert system described above using any commercially available expert system shell and inference engine. The syntax of CLIPS ("C" Language Integrated Production System) inference engine is used as a medium of description. The CLIPS inference engine has been developed at the NASA Johnson Space Flight Center and is currently marketed by COSMIC [16]. The objective of this description to demonstrate the methodology and not to provide an actual source code. The system is composed of the following components:

2.2.1 Fact List

The fact list is used to convey information about the current state of the network environment to the expert system. This information is obtained from the Management Information Base (MIB) maintained by the OSI protocols. The expert system uses this knowledge to compute the FRT at each site on the network and to detect any deterioration in performance. The fact list is also used by the expert system to determine possible options available to improve the performance. Some of the facts asserted by the MIB are described below:

(Sites Site(1) Site(2) Site(3) ...)
(Site(i) No_of_users Utilization)

The above facts provide the expert system with information about the active sites in the system as well as the number of users at each site (with respect to the particular application) and the utilization of each site. The fact,

(Application(A1) AS1 AS2 AS3 AS4 ...)

informs the expert system that application A1 requires the specified application servers (ASi). Information about each AS is conveyed to the expert system by the fact below.

(AS(i) Site Size Times_invoked Eff_service_time Utilization ...)

This fact informs the expert system about the location and size of each application server, the number of times it is invoked during one execution of the distributed application, the service time of the application server, its utilization, etc. Information about the effective transmission delay per packet on each communication link is conveyed by the following set of facts:

(T_{Delay} *LAN1 delay_time*)
(T_{Delay} *LAN2 delay_time*)
...
(T_{Delay} *FDDI delay_time*)

Other information can be conveyed to the expert system in a similar manner. This information is then used by the expert system knowledge base to evaluate the system performance and optimize it if necessary.

2.2.2 Knowledge Base

The knowledge base of an expert system consists of a set of of rules which define the expertise or intelligence of the expert system. The rules interpret the information provided by the fact list and decide on the course of action to be taken. CLIPS rules have an "if-then" format where the "if" part matches system information provided by the fact list and the "then" part defines the action to be taken. The following rule illustrates the concept.

```
(defrule Monitor_Traffic "Monitor the traffic on a network"
    (Traffic > 0.75)
    (Alternate-route Available)
⇒
    (Re-route Traffic)
)
```

The knowledge base for the expert system described in this paper consists of the following two modules:

Resource Monitor The resource monitor, described in section 2.1.2, is triggered at regular intervals and is provided with information regarding the current system state.

The monitor uses this information to evaluate the FRT at each site. The communication delay experienced by a client at a site in accessing an application server can be evaluated as the summation of the communication time over each link in the subnetwork connecting the client site to the server site. The list of communication links in a subnetwork from a given site i to a site j are stored by the system as a table. Thus the calculation of the communication delay involves mainly table lookup which can be performed efficiently using the pattern matching capabilities of the inference engine. A format for the routing fact table is shown in Table 1. Here, each entry in the table is a fact of the type:

(Net[i,j] Link)

where $Net[i,j]$ identifies the client site (i) and the server site (j), i.e. the row and column of the table respectively; while *Link* is a communication link in the subnetwork connecting i to j. If the subnetwork consists of more than one communication link, then the table will have one entry for each link in the subnetwork.

The communication delay can now be calculated using the following rule:

```
(defrule Comm "Calculate communication delay "
    (CALC DELAY Site?i AS(?j))
    (Net[?i,?j] ?LAN)
    ?f1 < − (T_Delay ?LAN ?delay)
    ?f2 < − (TOTAL_DELAY(?i) ?total_delay)
    (AS(?j) ? ?size ?times $?)
⇒
    (retract ?f1 ?f2)
    (bind ?net_delay (* ?delay ?size ?times))
    (bind ?total_delay (+ ?total_delay ?net_delay))
    (assert (TOTAL_DELAY(?i) ?total_delay))
)
```

The processing delay can similarly be calculated by using the effective service time information provided by the MIB. A set of rules to achieve this are listed below:

```
(defrule Proc_per_AS "Calculate processing delay per AS"
    ?f1 < − (CALC PROC AS(?j))
    (AS(?j) ? ? ?times ?service_time $?)
⇒
    (retract ?f1)
    (bind proc_time ( ?service_time ?times))
    (assert (PROC_TIME(?j) ?proc_time))
)
```

```
(defrule Proc "Calculate processing delay"
    (CALC PROC TOTAL)
    ?f1 < − (PROC_TIME(?) ?proc_time)
    ?f2 < − (TOTAL_PROC_TIME ?total_proc_time))
⇒
    (retract ?f1 ?f2)
    (bind proc_time (+ ?total_proc_time ?proc_time))
    (assert (TOTAL_PROC_TIME ?total_proc_time))
)
```

The function response time or FRT for each client site is now calculated using equation 1 as the sum of the communication and processing delays. The average FRT is then compared with the Max_Acc_FRT and if it is found to be greater, the Performance Optimizer is invoked. The following set of rules handle these tasks:

```
(defrule FRT "Calculate FRT per site & total FRT"
    ?f1 < − (TOTAL_DELAY(?i) ?total_delay)
    (TOTAL_PROC_TIME ?total_proc_time)
    ?f2 < − (TOTAL_FRT ?total_ft)
⇒
    (retract ?f1 ?f2)
    (bind ?frt (+ ?total_delay ?total_proc_time))
    (bind ?total_frt (+ ?total_frt ?frt))
    (assert (FRT[?i] ?frt))
    (assert (TOTAL_FRT ?total_frt))
)
```

```
(defrule Calc_Avg_FRT "Calculate average FRT"
    (Sites ?$sites)
```

	AS(1)	AS(2)	AS(3)	AS(4)	...
Site 1	(Net[1,1] LAN1)	(Net[1,2] LAN1)	(Net[1,3] LAN1) (Net[1,3] FDDI) (Net[1,3] LAN3)	(Net[1,4] LAN1) (Net[1,4] FDDI) (Net[1,4] LAN3)	...
Site 2	(Net[2,1] LAN2) (Net[2,1] FDDI) (Net[2,1] LAN1)	(Net[2,2] LAN2) (Net[2,2] FDDI) (Net[2,2] LAN1)	(Net[2,3] LAN2) (Net[2,3] FDDI) (Net[2,3] LAN3)	(Net[2,4] LAN2) (Net[2,4] FDDI) (Net[2,4] LAN3)	...
Site 3	(Net[3,1] LAN3) (Net[3,1] FDDI) (Net[3,1] LAN1)	(Net[3,2] LAN3) (Net[3,2] FDDI) (Net[3,2] LAN1)	(Net[3,3] LAN3)	(Net[3,4] LAN3)	...

Table 1: Routing Fact Table

```
    (TOTAL_FRT ?total_frt)
 ⇒
    (bind ?num_sites (length ?$sites))
    (bind ?avg_frt (/ ?total_frt ?num_sites))
    (assert (Avg_FRT ?avg_frt))
 )
```

```
 (defrule Trig_Opt “Trigger optimizer if necessary”
    (Max_Acc_FRT ?max_acc_frt)
    (Avg_FRT ?avg_frt&:( > ?avg_frt ?max_acc_frt))
 ⇒
    (KICKOFF PERFORMANCE_OPTIMIZER)
 )
```

Performance Optimizer The Performance Optimizer is triggered by the Resource Monitor when an unacceptable FRT is detected i.e. when CurAvg_FRT is greater than Max_Acc_FRT. The Performance Optimizer first saves the current configuration and current average FRT which is accomplished by the following rules:

```
 (defrule Save_Cur_Config “Save the current configuration”
    (AS[?i] ?$data)
 ⇒
    (assert (Cur_Config AS[?i] ?$data))
 )
```

```
 (defrule Save_Avg_FRT “Save the current average FRT”
    (CurAvg_FRT ?avg_rt)
 ⇒
    (assert (Cur_Avg ?avg_frt))
 )
```

It then identifies the the site with the largest FRT and tries to improve it by moving the smallest migratable resource from the site with the minimum FRT (MIG_FROM_SITE) to the site with the maximum FRT (MIG_TO_SITE). Migration can be accomplished by the following rules:

```
 (defrule Migrate “Migrate resource”
    (MIG_FROM_SITE ?mig_from_site)
    (MIG_TO_SITE ?mig_to_site)
    (MIGRATABLE_AS ?mig_as)
    ?f1 < − (AS[?mig_as] ?mig_from_site ?size ?times ?$data)
 ⇒
    (retract ?f1)
    (assert (AS[?mig_as] ?mig_to_site ?size ?times ?$data))
 )
```

The recalculation of the FRT's can be done using the rules used by the Resource Monitor. If the above migration improves the average FRT, i.e. the NewAvg_FRT is less than CurAvg_FRT, it is accepted and the values of the CurAvg_FRT and Cur_Config FRT are updated using the rules defined above. If there is no improvement in performance, the current migration is discarded and the next possible migration is activated by appropriately defining the values of MIG_FROM_SITE and MIG_TO_SITE.

2.2.3 Inference Engine

The inference engine provides the pattern matching capability allowing the knowledge base to be triggered by the fact list. In the implementation presented in this section we use the CLIPS forward chaining inference engine. CLIPS complements the forward chaining nature of the application and provides efficient “C” routines which are easy to interface with.

2.3 An Illustrative Example

In this subsection we present a numerical example to illustrate the operation of the expert system

Hence decision taken: *Migrate AS1 from Site1 to Site2*
The traffic on each network for the new configuration can be calculated using equation 8 and are found to be: $\rho_{LAN1} = 10.31\%$; $\rho_{LAN2} = 12.93\%$; $\rho_{LAN3} = 13.2\%$; $\rho_{FDDI} = 1.59\%$. The new FRT's can now be calculated. They are:

$$\begin{array}{l} FRT_1 = 7.353 \text{ sec} \\ FRT_2 = 7.764 \text{ sec} \\ FRT_3 = 6.834 \text{ sec} \\ \text{New\_Avg} = 7.317 \\ \text{Min. FRT} = \text{Site 3} \\ \text{Max. FRT} = \text{Site 2} \end{array}$$

Hence decision taken: *Migrate AS3 from Site3 to Site2*
Note that the $NewAvg\_FRT < CurAvg\_FRT$ and hence we assign $NewAvg\_FRT$ to $CurAvg\_FRT$ and assign the configuration to Cur_Config

The new traffic on the networks are found to be: $\rho_{LAN1} = 10.3\%$; $\rho_{LAN2} = 14.0\%$; $\rho_{LAN3} = 11.55\%$; $\rho_{FDDI} = 1.53\%$. The new FRT's can now be calculated. They are found to be:

$$\begin{array}{l} FRT_1 = 7.309 \text{ sec} \\ FRT_2 = 7.393 \text{ sec} \\ FRT_3 = 7.091 \text{ sec} \\ \text{Current\_Avg} = 7.264 \\ \text{Min. FRT} = \text{Site 3} \\ \text{Max. FRT} = \text{Site 2} \end{array}$$

Note again that the $NewAvg\_FRT < CurAvg\_FRT$ and hence we assign $NewAvg\_FRT$ to $CurAvg\_FRT$ and the configuration to Cur_Config The procedure continues until the specified convergence criterion is met.

3 Summary and Concluding Remarks

In this paper, we presented a general architecture of an expert system to manage the performance of a distributed application whose application servers and clients are geographically distributed across several local area The system monitors the performance experienced at different user sites and redistributes the resources when a deterioration in performance is encountered. The redistribution attempts to optimize two parameters; (1) the average FRT across the system and (2) the variance of the FRT across various client sites. We develop, a set of rules and evaluation procedures to estimate the functional response times at each client site. We are currently adopting the proposed expert system based performance optimizer to achieve load balancing in a heterogeneous environment consisting of parallel computers workstations. Further research is needed to extend the functionality of the proposed system to handle all the functions of the ISO network management system (viz. configuration management, fault management, accounting management and security management).

References

[1] K. H. Muralidhar and B. W. Irish, "MAPCON: An Expert System for Configuration of MAP Networks," *IEEE Journal on Selected Areas of Communications* Vol. 6, No. 5, June 1988.

[2] Mary Jane Strohl, "High Performance Distributed Computing in FDDI," *IEELTS*, May 1991, pp. 11-15.

[3] W. Bux , "Performance issues in local-area networks", *IBM Systems Journal,* Vol. 23, No. 4, 1984.

[4] W. Bux, "Local-Area Subnetworks: A Performance Comparison," *IEEE Trans. on Comm.*, vol. COM-29, no. 10, Oct. 1981 1465-1473.

[5] J. Laprie, "Dependable computing and fault tolerance: concepts and terminology", *Proc. 15th. Fault tolerant computing symposium,* Ann Arbour, Michigan, June 1985, p2-11.

[6] M. SLOMAN, "Distributed systems management", *Issues in LAN management. Proc. of the IFIP/W6.4A workshop*, Berlin 1988.

[7] ISO, "International Standard ISO 7498 Part 4 - OSI Management Framework", ISO 7498/Part 4, 1986.

[8] D. Kanyuh, "An integrated network management product", *IBM systems journal,* Vol. 27, No. 1, 1988.

[9] L. Kleinrock, *Queuing Systems, Volume II: Computer Applications.* New York: Wiley, 1976.

[10] L. Bernstein and C. M. Yuhas, "Expert Systems in Network Management - The Second Revolution," *IEEE Journal on Selected Areas in Communications,* June 1988, pp. 784-787.

[11] T. Yamaihra, Y. Kiriha, and S. Sakata, “Network Troubleshooting Expert System EXNETS”, *NEC Res & Develop.,* No. 94, July 1989, pp. 120-128.

[12] S. Ceri, L. Tanca, “Expert Design of Local Area Networks”, *IEEE Expert*, October 1990, pp. 23-33.

[13] A. S. Tanenbaum, *Computer Networks,* Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1988.

[14] K. Chae and A. A. Nilsson, “Performance Evaluation of FDDI Networks and Interconnected Heterogeneous Networks”, *IEEE, 1990.*

[15] Greg Chesson, The Protocol Engine Project, In *Proceedings of the Summer 1987 USENIX Conference*, pages 209–215, June 1987.

[16] Joseph C. Giarratano, ĈLIPS Users's Guide, Artificial Intelligence Section, Lyndon B. Johnson Space Center, October 1989.