マルチメディア通信と分散処理 59−11
（１９９３. １. ２９）

分散協調環境 Noah (Network oriented applications harmony) の提案

小塚 宏　佐藤 文明　宮崎 一哉　福岡 久雄

三菱電機 (株) 情報電子研究所

Noah(Network oriented applications harmony) は、場を介した通信機構上に協調プロトコル及び、アプリケーションとシステムの動作状態を監視／制御する機構を提供することによって、分散処理環境上で複数のアプリケーション間の協調動作を支援するためのソフトウェア・プラットフォームである。

Noah では、“協調 (applications harmony)” という言葉を次のように定義する。

能動的に動作する複数のアプリケーションあるいはアプリケーションの機能単位が、ある動作環境 (“場”) を通して相互作用することによって、場に定義された制約を解決する方向に自己の動作を制御すること。

場と協調機構そのものに継承機能を持つ階層構造を提供することにより、協調の方式、度合、範囲を柔軟かつ容易に設定することができる。

An Architecture of Distributed Computing Environment — Noah: Network Oriented Applications Harmony

Hiroshi KOZUKA, Fumiaki SATO, Kazuya MIYAZAKI, Hisao FUKUOKA

Computer & Information Systems Laboratory
Mitsubishi Electric Corporation
5-1-1, Ofuna, Kamakura, Kanagawa 247, Japan

Noah (Network oriented applications harmony) is a software platform to support distributed applications cooperation, and provides communication mechanisms based on cooperation field, cooperation protocols, agents and system management / control procedures.

In the Noah architecture, we define the term “applications harmony” as the following.

“Applications harmony” means an application control mechanism to solve some constraints defined in the “field” (operational environment) at which applications or functional units work and interact autonomously.

By having the inheritance mechanism in the communication field and cooperation function, Noah is able to configure the method and the scope of the applications harmony flexibly.

1 Background and Objectives

Distributed computing system, where several networked computers work cooperatively, has been widely spreading under the wave of down-sizing. In such environments, high cost performance and reliable services are expected. On the other hand, the distributed computing system brings several new problems that a conventional centralized system does not have.

- System administration :
  In distributed environment, management tasks of addresses, names, peripheral devices, and networks are much more complex than in centralized systems.
- End user's operation :
  As resources and services are distributed, end users have to be conscious of their physical locations. Furthermore, it is necessary to change the commands or sequences of operation on the resources and the services according to their locations.
- Application development :
  In order to describe applications that effectively use the distributed environment, it is necessary to distribute functions of the applications and to program some complex processes such as communications among distributed modules, synchronization of them, and access control of resources. Besides, to add more values such as concurrency of the modules and multiplicity of the functions for improvement of the reliability, programming tasks are extremely complicated.

To cope with these problems, some countermeasures are taken. On the system management of distributed environment, systems that concentrately manage resources using management information sent by system management processes stationed at each computer are constructed. But such systems are very weak for a damage of the central management system.

On the distributed application development, the development environments such as distributed object-oriented languages succeed in giving some transparency of an access to networks and resources to the users. But in conventional environments, relations between objects are fixed, so it is impossible to add new services flexibly. Also it is impossible to change the way to provide the services.

Based on the background described above, we proposed Noah, the cooperation mechanism among applications in distributed computing environments. The major objectives of Noah are to provide followings:

- Programming environment for autonomous program modules, agents.
- Cooperative mechanism for multiple agents.
- Communication mechanism for distributed agents.
- Management mechanism to control agents.

These features improve fault tolerance of systems and availability of services, and make it easy to develop reliable and efficient softwares using potentiality of services of distributed environment.

Section 2 shows the concept of Noah, section 3 describes the Noah architecture in detail, section 4 explains the services that Noah can provide and the outline of some applications' behavior on Noah, and section 5 presents the summary and future works.

2 Concept of Noah

Here describes the basic concept of Noah – Network oriented applications harmony. Noah is a software platform for cooperation mechanism of applications in distributed computing environment.

2.1 Applications harmony

"Applications harmony" means an application control mechanism to solve some constraints defined in the "field" (operational environment) at which applications or functional units work and interact autonomously.

"Cooperation field" consists of a goal, an evaluation criteria to measure a distance to the goal, and agents that achieve the goal cooperatively. The cooperation field specifies a cooperation process for agents to achieve the goal. According to the specified process, agents work cooperatively. In addition,the cooperation field specifies the members of agents joining some cooperational

tasks, qualifications for the membership, and the process for the participation.

The term "agent" or "cooperation" has different meanings since it has been used in many different contexts. Nakajima et al.[1] defines these terms as follows:

Agent: An agent is a program unit which has a certain capability. Although it is similar to the concept of a module, it does not imply information hiding. An agent is not necessarily intelligent. Contrary to the traditional concept of a program unit that performs a predefined task upon invocation, an agent works autonomously to some extent.

Cooperation: Cooperation means consistent behavior of agents. It includes cooperation, negotiation, compromise and so on.

2.2 Conceptual Model of Noah — Lake/Canal Model

The fundamental concept of Noah computing architecture is based on the Lake/Canal model shown in Fig.1. In this model, a distributed application consists of several active processing units, i.e. agents, and each agent takes actions to its environment to proceed on its task. Basically, the Lake/Canal model improves the Linda model[2] developed at Yale University.

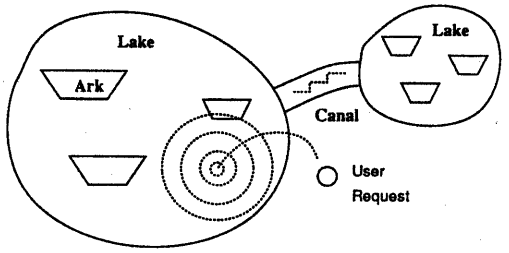

Figure 1: Lake/Canal Model

The Lake/Canal model consists of the following abstract entities.

Lake: A Lake is the field where the communication among agents takes place. A set of computers of the same architecture forms a Lake.

Canal: A Canal is the gateway which interconnects two Lakes. It performs the data/message transformation to connect Lakes of different architecture.

Ark: An Ark is a function unit of an application and can be regarded as being on a Lake. It is an active agent, which performs its task sensing the status change of the Lake and let the Lake transit to the other state by giving the processing results to it.

Applications in Noah can cooperate by communicating via the Lake, the communication field. Since agents, which make up an application, can perform their own tasks autonomously to some extent, the cooperation can be achieved in a relatively easy manner.

Fig.2 shows the place of Noah in a layered software architecture as a vertical section of "Lake Noah."

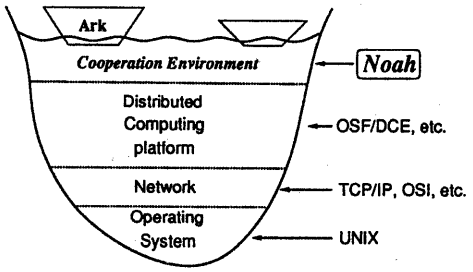

Figure 2: Vertical section of Lake Noah

3 Noah environment

3.1 Architecture

The Noah architecture has the following features.

- Hierarchical cooperation fields are provided for the communication among agents.
- Each cooperation field supports inheritance of the cooperation mechanism.

Having the inheritance mechanism in the communication field and cooperation function, Noah is able to configure the method and the scope of the applications harmony flexibly.

To realize this features, Noah architecture has the following components (see Fig. 3).

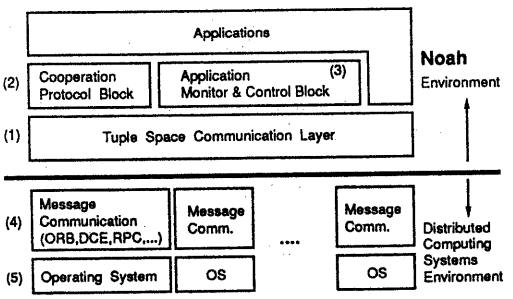

Figure 3: Noah architecture

(1) Tuple Space Communication Layer:
Tuple space communication layer provides synchronization, data sharing and communication among agents.

(2) Cooperation Protocol Block:
Cooperation protocol block provides some common facilities to perform various types of cooperation among agents.

(3) Application Monitor & Control Block:
Application monitor & control block monitors and controls agents. The control means invocation and termination of applications, management of replication, modification of attributes, and so on.

(4) Message Communication:
Message communication (RPC, ORB, socket and so on.) provides the basic communication mechanism for tuple space communication.

(5) Operating System:
Operating system such as UNIX.

(1) $\sim$ (3) are provided by Noah environment. (4) and (5) will be available as de facto standard distributed computing system environment, for example, OSF/DCE, UNIX operating system and so on.

One of the main features of the Noah architecture is that the communication mechanism (computational model), cooperation protocols and application control mechanisms are fully integrated. In the traditional research, each module has been considered separately. In addition the communication models are based on peer to peer communication. On the contrary, agents of the Noah architecture cooperate with each other watching the cooperation field.

Noah's computational model using hierarchical fields makes it easier to set the scope of the cooperation and the implementation on the distributed computing system.

3.2 Components

3.2.1 Tuple space communication layer

This layer provides transparent tuple space using lower message communication layer. This layer supports transparency for remote node access and remote tuple space management.

Tuple space was originally developed as the communication medium of Linda. The process uses very simple interaction mechanism with tuple space and it can communicate with other multiple processes simultaneously. Tuple space of Linda can be regarded as a shared memory among processes. It is very useful communication mechanism, but it also has some problems, such as the dead lock or conflict caused by errors and collisions of tuple identifiers. Therefore, the layering and structuring of the tuple space is under further research[3].

In Noah's tuple space communication, any data exchanged among agents is treated as tuple. Agents communicate with each other by putting/getting tuples to/from the tuple space. Unlike RPC or object oriented message passing mechanisms, tuple space communication does not specify the receiver of the message. So it is very useful for broadcast, multicast and asynchronous communications.

The communication mechanism with tuple space is suitable the concept of Noah — applications harmony, because agents in the Noah environment take part in the cooperation field autonomously.

Noah uses an extended Linda model, which we call "**LAKE**" model. The LAKE model has a computational entity called "**Ark**" and a small tuple space. Both the Ark and the tuple space compose a "**Lake.**" Applications are composed by multiple Lakes. The Lake may create another Lake and purge it. The Ark can interact with the tuple space identified by the argument Lake. The feature of our LAKE computational model is a shared tuple space between a parent Ark and its child Ark. The communication from the outside of the application to the parent Lake influences the child Lake implicitly. Therefore, the applica-

tion can cooperate with others tightly. Of course, LAKE also has a facility to create the Lake isolated with parent Lake. Fig.4 indicates the relation of tuple space of the parent and child Lake.

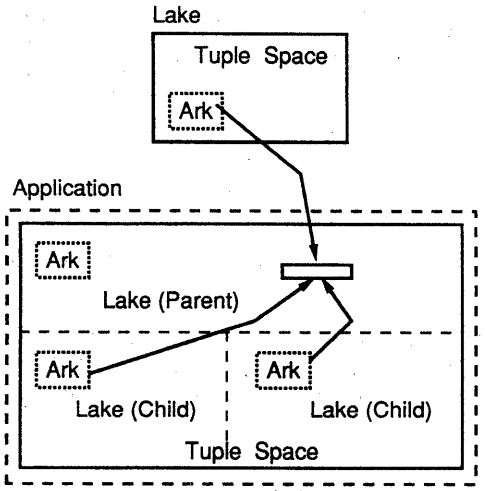

Figure 4: LAKE model

3.2.2 Cooperation protocol block

To perform the cooperation among applications, many functions should be executed — choice of the application to cooperate, selection of the cooperation protocol, data exchange among applications, estimation of the result caused from applications' cooperation, and so on. If we provide these procedures to each application packages, efficiency of the software development is low, and expansion or configuration of the system will not be achieved easily. So we concentrate the common functions to be used in cooperative communication in a single cooperation protocol block. This approach will result in an efficient software development and easy customizing of applications.

For example, Contract net[4] is one of the basic cooperation protocols. It is a procedure to invite servers by sending a request specification to them. User can select the most suitable server by receiving response specifications from the servers and can ask the server to do the request.

In Noah, many other cooperation protocols will be available, for example, Multistage negotiation[5], Hierarchical protocol[6] and so on. In addition, time dependent constraint, scope of cooperation field are usable as optional specification. Furthermore, the cooperation protocol block provides some mechanisms to establish a cooperation field which can creates hierarchical cooperation fields for each applications.

3.2.3 Application monitor & control block

Sometimes dynamic invocation and termination of a particular agent (application), or the modification of the agent's (application's) behavior and monitor are required to reuse the software developed in other environment. Noah can support those requirements using the function of application monitor & control block. This block provides the invocation, termination, dynamic attribute modification of agents, and monitor of the state of agents and systems.

Noah provides "**Probe**" and "**Activator**" which are extensions of sensor and actuator in Meta[7] developed at Cornell University respectively. Probe and activator are the door to monitor and control interface of agent. Monitor and control request are processed through probe and activator. The relation (external structure) of agents is shown in Fig.5.

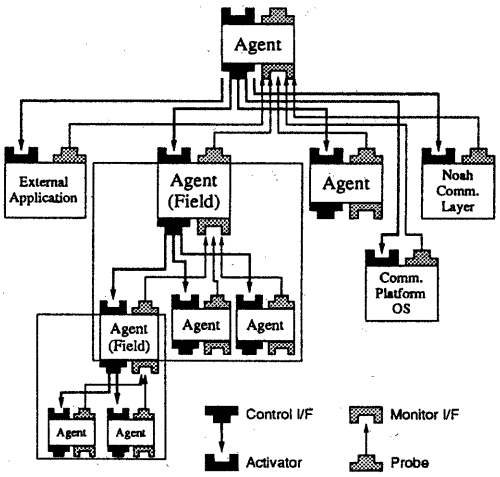

Figure 5: External structure of agent

The probe and activator can be constructed or attached to the agent. Prove get the state of agent internal data from shared memory space (tuple space) and send it to monitor interface periodically. Activator receive the messages from control interface and put it into tuple space. If agents

in Noah environment have hierarchy, parent agent can monitor and control it's children with probe and activator.

4 Applications

Noah cooperative environment allows applications to cooperate in different styles. This section describes example of typical cooperation styles to show the advantages of Noah.

4.1 System administration

Here is an example of a printer installation into distributed computing environment. This is a typical system administration issue.

In a traditional distributed computing environment such as UNIX based network environment, a system manager needs the following operations to install a new printer into that environment.

- For printer installed network node.
  - Add the new network node into host name database.
  - Add the new printer information into printer capability database.
  - Set up the printer spool directory.
  - Create printer access control file.
  - Modify the equipment database.
- For other network node.
  - Add new printer information into printer capability data base.
  - Notification of new printer information to users.

These operations are very complicated and should be consistent. So the system manager is apt to mistake the operations.

While using Noah environment, what the system manager have to do is just a putting "new printer agent" into Noah distributed environment. It works autonomously in the Lake, and does the above operations concurrently and consistently as shown in Fig.6.

Because new printer agent will modify the Lake status, all agents related to printer installation — such as printcap_db agent, hosts_db agents, messaging_agents — are works by themselves.

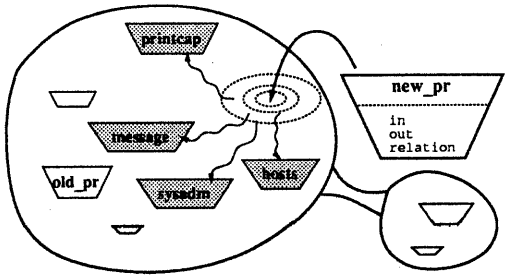

Figure 6: Installation of new printer

4.2 Personal information management

Conventionally, each CSCW (Computer Supported Cooperative Works) application manages groups and its members information individually, and supports communications among the members (Fig.7).

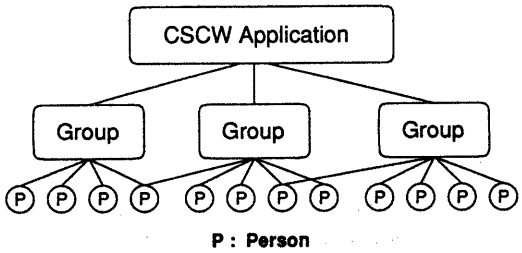

Figure 7: Conventional CSCW Application

We propose that the same functionality can be attained by using cooperating personal applications in conjunction with the open CSCW environment on top of the Noah technology (Fig.8). Using the open CSCW environment, it is possible to centralize the group management functions and use them for several CSCW services.

The open CSCW environment can be realized as 'a field' of Noah. 'Groups' in the open CSCW environment can also be mapped to a Noah field. Other OA (Office Automation) applications such as a meeting room reservation system or a meeting calling system, can cooperate with CSCW applications through another 'field'.

The open CSCW environment allows these fields to construct CSCW services adaptable to organizational changes and to cooperate with many other services easily. Fig.9 shows the CSCW system structure.

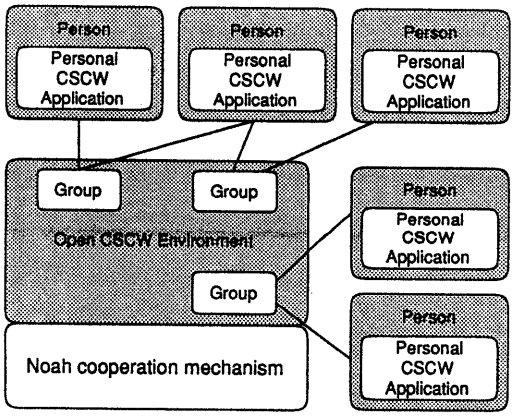

Figure 8: Open CSCW environment

CSCW manager — The CSCW manager creates a CSCW field and manages it. In this field, the CSCW manager realizes CSCW specific cooperation. The main functions of this field include organization management (management of group structures, locations of members, and roles of members in the groups), data sharing, access control, and work flow supports.

OA manager — The OA manager creates an OA field and manages it. In this field, OA applications and personal information systems work together harmoniously.

Personal Information System — The Personal information system maintains a personal profile and personal schedules for an individual. This system includes an access interface to the Noah environment.

Other Applications — OA services, such as the meeting calling system and the meeting room reservation system, are provided.

When personal information system is connected to the Noah environment, the personal profile key by personal information system is broadcasted in the Noah environment. Then the CSCW manager accepts the profile and keeps it. The profile includes the location of the personal information system and attributes of the user (group names the user belongs to and the role of the user in the groups).

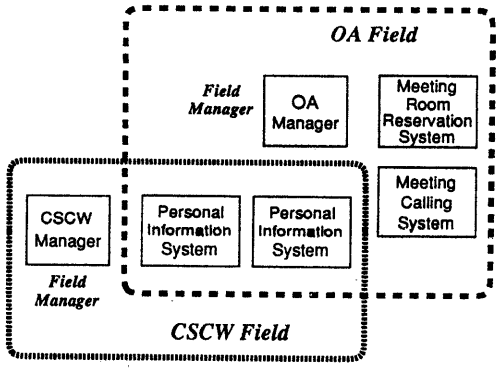

Figure 9: Noah CSCW system structure

Now let us assume the case of holding a meeting with members of certain groups. The following is an example of a service for this case.

1. The organizer of the meeting requests the confirmation of members schedules and the reservation of the meeting room. For this, attending groups, date, time interval, and characteristics of the meeting room are specified through the user access environment of the personal information system.

2. The request is sent to the CSCW manager and the OA manager through the Noah environment.

3. CSCW Manager:

   (a) After accepting the request, the CSCW manager inquires the personal information system of the specified groups' members about their unoccupied hours in the corresponding of time interval and date.

   (b) The personal information systems that accept the request return the information about the unoccupied hours according to their own schedules.

   (c) Then the CSCW manager arranges the information in order and send it back to the organizer.

4. OA Manager:

   (a) After accepting the request, the OA manager requires the meeting room reservation system to inform of the

unoccupied hours of the appropriate rooms.

(b) The meeting room reservation system returns the information to the OA manager and the manager sends it to the organizer.

5. The organizer decides the date, the time and the room based on the information and sends the CSCW manager and the OA manager the final decision to hold the meeting.

6. After accepting the request, the CSCW manager informs the members of the meeting and the OA manager requests the meeting room reservation system to reserve the specified room at the specified time.

Since the information about organizations can be centralized and can be collected automatically, the system can be adapted to organizations changes.

5 Summary and Future Works

This paper proposed Noah: Network oriented applications harmony as an architecture of distributed environment. Noah is suitable for resource allocation, fault-tolerant applications, system management, and applications for cooperative works. Noah provides following functions.

- Programming environment for autonomous agents.
- Cooperating mechanism for multiple agents.
- Communication mechanism for distributed agents.
- Control and management mechanism for agents.

Noah makes it easy to develop reliable and efficient softwares using potentiality of services of distributed environment.

Design of programming interface, language and development environment are future work. Design and development of a "shell" to manage interactions between users and the environment are also future works.

References

[1] Nakajima et al. "Current Status of Researches on Cooperative Architecture," (in Japanese), Electorotechnical Laboratory Research Report No.221, 1991.

[2] Sudhir Ahuja, Nicholas Carriero, David Gelernter, "Linda and Friends," IEEE COMPUTER, Vol.19, No.8, Aug.1986.

[3] Yoshida,N., Narazaki,S., "A Parallel Cooperation Model 'Cellula' Composed of 'Process + Field' Amalgams"(in Japanese), J. IPSJ 31:7(1990)1071-1079.

[4] R. G. Smith, "The contract net protocol: high level communication and control in a distributed problem solver", IEEE Trans. on Comput., vol. C-29, no. 12, pp. 1104-1113, 1980.

[5] K. Kuwabara and V. R. Lesser, "Extended protocol for multistage negotiation", Proc. 9th Workshop on Distributed Artificial Intelligence, 1989.

[6] E. H. Durfee and T. A. Montgomery, "A hierarchical protocol for coordinating multiagent behaviors", Proc. AAAI-90(1990).

[7] Keith Marzullo, Robert Cooper, Mark D. Wood, Kenneth P. Birman, "Tools for Distributed Application Management," IEEE COMPUTER, Vol.24, No.8, Aug.1991.