

# Agent-oriented Modeling Approach for Distributed Network Management Applications

Peng Zhang\*<sup>a</sup>, Cong Zhao\*\*<sup>b</sup>, Zengzhi Li\*\*\*<sup>c</sup>

<sup>a</sup>Bell Labs Research China, Lucent Technologies; <sup>b</sup>Neocomputer Research Institute, Xi'an Jiaotong Univ.; <sup>c</sup>Institute of Computer Architecture & Networks, Xi'an Jiaotong Univ.;

## ABSTRACT

In recent years, software agent has gained great momentum in network management, and it is recognized as an effective technology to build distributed network management system. However, most researchers focus their attentions on the distributed management applications based on some specific agent architectures or platforms. Up to now, there is no a common methodology to guide the construction of the system of this paradigm. From the viewpoint of modeling, this paper adopts multiple models used in software agent and combines them together to form a novel systematical modeling approach. This modeling approach covers two stages in development: system analysis and system design. A prototype for managing network flows was also developed to testify the modeling approach.

**Keywords:** distributed network management, multi-agent system, software agent, distributed system, network flow

## 1. INTRODUCTION

Most network management systems of today are organized according to the centralized paradigm, where the intelligence and control are placed in one centralized Network Management Station (NMS), or the platform-centric paradigm, where network management applications are built on a single management platform. The two paradigms can largely solve the problems of interconnection and interoperation between different management components. However, the lack of scalability, flexibility and reliability makes both the paradigms unable to manage large heterogeneous networks effectively (especially when the network size increasing continuously like today's Internet). Under this circumstance, the paradigm of distributed network management was put forward as a potential solution, in which the management functions is no longer centralized, but distributed into multiple management entities all over the network.

Compared with traditional paradigms, the distributed paradigm provides a stronger scalability, flexibility and reliability, but it also brings some new challenges. Among them the key problem is how to construct such a kind of network management system. Generally speaking, the complexity of a distributed system is far beyond that of a centralized system. In the network management context, things are the same. Up till now, there is no a commonly recognized approach able to be used for setting up a distributed network management system.

In recent years, many researchers attempt to employ varieties of new technologies for developing distributed network management systems, such as Distributed Objects, Component technology, WEB, Active Network, etc <sup>[1]</sup>. Among them, Software Agent Technology (SAT) has become the focus of all the attentions. Software agent originated from distributed artificial intelligence (DAI)<sup>[1][2]</sup>, and has some promising features very suitable for solving the problems brought by distributed systems, such as autonomy, reactivity, social ability, proactiveness, etc. Unfortunately, because software agent technology has not been fully developed, its research on network management still stays at the initial stage and needs a period of time to become mature.

The goal of this paper is to provide a novel modeling approach for distributed network management by using software agent. The distributed network management system here consists of a number of relatively small functional entities, agents, which can negotiate with each other and coordinate their behaviors automatically so that the network management tasks can be accomplished in a cooperative way. Actually, this kind of network management system is a multi-agent system. The modeling approach is composed of a series of models organized according to the sequence of system development, which cover two stages in system development, requirement analysis and system design. Because the implementation stage is not included in this approach, it is a "modeling approach" rather than a "development methodology". But the result of this model-driven approach provides a good foundation for the following implementation stage.

The following part of this paper is structured into four sections. Section II gives an overview for the modeling approaches. Section III focuses on the goal model and role model used in the stage of requirements analysis. Task Model, agent model and information model proposed for system design are discussed in section IV. The emphasis of the two sections is on how to modeling the network management application with these models. Except for information model, all the examples for other models are developed for a prototype system to manage Multi-path network flows. In the last two sections, the prototype is introduced and the conclusion is made respectively.

## 2. AN OVERVIEW FOR THE AGENT-BASED MODELING APPROACH

This modeling approach is based on agent-oriented software engineering (AOSE). The concept of AOSE was first put forward by Jennings in 1998<sup>[3][4]</sup>. As a newly emerging field, AOSE has no standardized methodology for the development of multi-agent system. Instead, a number of concepts and approaches from other fields have been absorbed by AOSE. For example, Wooldridge proposed the agent-oriented analysis and design on the basis of Object-Oriented technology<sup>[5]</sup>; Kendall presented the agent-oriented role modeling technology that is an extension of the role concept in OO<sup>[6]</sup>; the project CommonKADS provided a multi-agent system modeling approach from the viewpoint of knowledge engineering<sup>[7]</sup>; Deloach extended universal modeling language (UML) and formed a systematical approach in the analysis and design of multi-agent system<sup>[8]</sup>. A seemingly common approach is to use OO principles to define the external models for agent-based system and internal models for an individual agent.

On the basis of the existing approaches, this paper adopts some models and extends them to form a novel multi-agent modeling technology for distributed network management. Five models are contained in this approach: goal model, role model, task model, agent model and information model. These models solve the problems that might be met in distributed

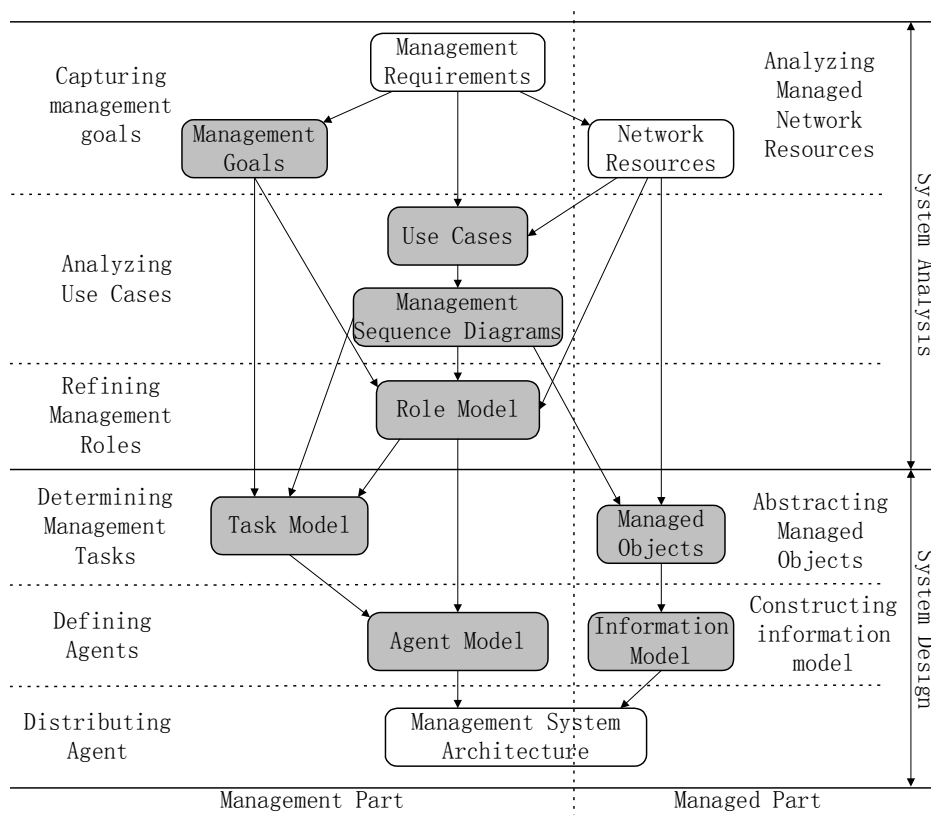


Figure 1 The Modeling Approach

network management, such as negotiation, coordination and cooperation. These models are mainly used in system analysis and system design, which makes the system development have the feature of “model-driven”. At the same time, the approach is not limited to network management, but also provides some useful guidance for multi-agent based systems.

The modeling processes are shown as Figure 1. From the viewpoint of management, a network management system consists of two parts, management part and managed part, as that was shown in Figure 1 and separated by the vertical dotted line. The blocks in gray are the models/tools used in modeling approach, and the arrows represent the transitions between these models/tools. With the system developing, these models give a more and more detail description towards the target system. The modeling approach contains several stages separated by dotted horizontal lines, and the main tasks for each stage are noted on both sides of the figure. The information model from general network management system is still kept in the figure so that the interfaces between agents and network resources can be simplified. It should be emphasized that the whole modeling approach is not a strictly sequential process, but an iterative process. The later modeling cycle is always much more detail than the former cycle.

This modeling approach is independent on some specific agent’s internal structures, programming language or message transmitting system. That is to say, the multi-agent system developed through this approach is convenient to be integrated and interconnected with other legacy systems. More details of this modeling approach will be described in the following sections.

### 3. SYSTEM ANALYSIS

The models involved in this stage are goal model and role model. The results of this stage will be the requirements specification of network management described by the two models.

#### 3.1. Management Goals Modeling

The first step in analysis is capturing management goals or goal modeling, which takes the initial system specification and transforms it into a structured set of system goals as shown in a Goal Hierarchy Diagram. A goal can be defined as a system-level objective, which represents some certain system intent. Here the concept of Goal Hierarchy Diagram is from some research works by Kenall [6] and Wood [8].

There are two tasks in this step: identifying and structuring goals. The goals may be included in the management requirements from network administrators, some detailed technical documents, or some specific management polices. Identifying goals is to abstract the essential part from these requirements description and list them as management goals.

The other task is to structure the goals in a Goal Hierarchy Diagram according to their importance and scope. Generally, higher-level goals in a Goal Hierarchy Diagram always contain the lower level goals and have a higher importance. Goals in the same level are roughly equal in scope and usually related to each other.

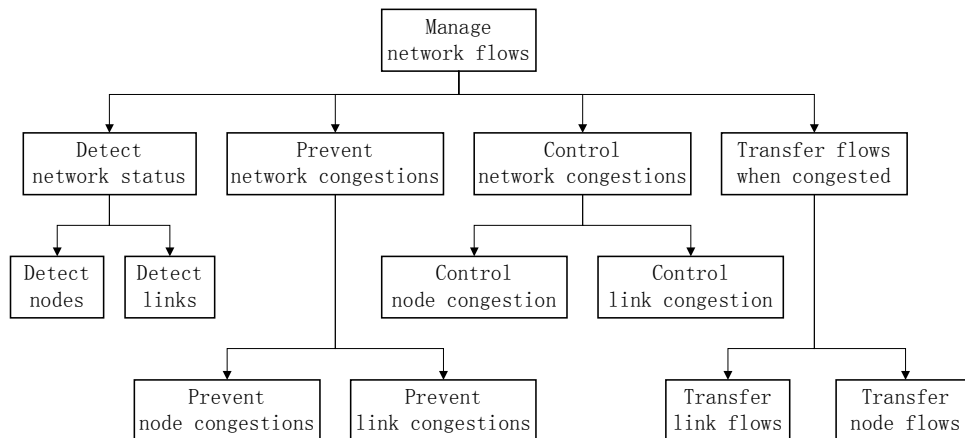


Figure 2 Goal Hierarchy Diagram

Figure 2 is the Goal Hierarchy Diagram that has been used in a network flows management prototype. This diagram is simply a three-level hierarchical structure.

### 3.2. Capturing Use Cases

“A use case is a sequence of transactions performed by a system that yields a measurable result of values for a particular actor”<sup>[9]</sup>. The concept use case firstly came from the field of Object-Oriented Software Engineering (OOSE) and has been adopted by unified modeling language (UML) as an important analysis tool <sup>[9]</sup>. Scenario, or the instance for a use case, is a realization for one use case, which is generally described by a sequence of events among different participants.

Capturing use cases is the second step in system analysis stage. The main tasks are to obtain use cases from system requirements and to depict their scenarios by message sequence diagrams. There are two kinds of components in a message sequence diagram: roles and messages. Role represents a specific position of an agent in the system, which is always corresponding to some functions or responsibilities. Messages exchanged between multiple roles describe the process of information transferred between different agents for accomplishing some network management functions.

If a message is passed between two roles, then there must exist a conversation between the two agents that play the roles. The agent that begins the conversation plays the initiator role, while the agent on the other side of the conversation becomes the responder.

Figure 3 is an example for message sequence diagram, which describes the scenario of balancing network flows among multiple output links connected to the same network node. There are five roles in this scenario, which belong to two types: initiator and responder. The initiator in Figure 3 is the link role (Link A) that sends out the load-balancing proposal. Responders here consist of coordinator and other links. Coordinator receives requests from multiple initiators, decides which proposal should be admitted and then sends back the accept message. With the requests from the link initiator, other links will count their free bandwidths and negotiate with the initiator to determine how much bandwidth they can provide for balancing network loads among links.

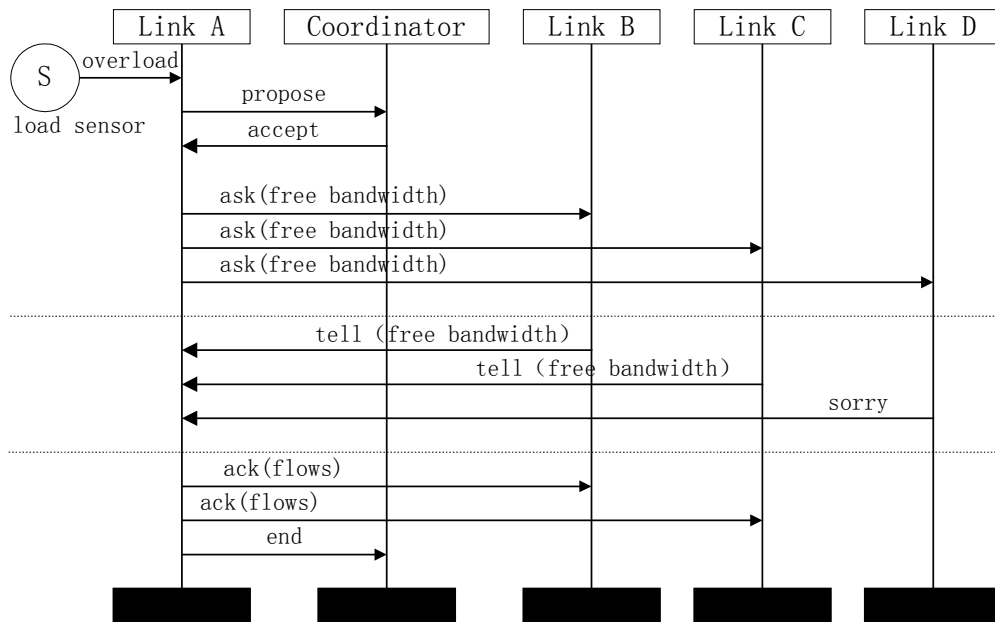


Figure 3 An example for message sequence chart

### 3.3. Role Modeling

The task of role modeling is to transform the structured goals of the Goal Hierarchy Diagram into a form more useful for constructing multi-agent based network management systems: roles. A role is an abstracted description for an entity's expected functions and encapsulates the system goals assigned to the entity<sup>[8]</sup>. Roles are the building blocks used to define agent's classes and to capture system goals during the design stage.

In general case the transformation from goals to roles is one-to-one: each goal maps to a role. However, there are many exceptional situations where it is more useful to combine goals. Similar or related goals may be combined into single roles for the sake of convenience or efficiency. Goals that share a high degree of cohesion can be combined into a single role.

Role definitions are captured in a Role Model as shown in Figure 4. There are two kinds of components in a role model diagram: roles and communications paths. Roles are represented as blocks that contain some role-related goals, and lines between roles denote communications paths. These paths are derived from the Sequence Diagrams developed in the previous step.

A part of the role model used in the network flows management prototype is shown in Figure 4. There are four roles in the diagram: node roles and link roles that respectively represent the corresponding network elements; administrator interface denotes the interaction function between the network administrator and the other parts of the system; the last role, coordinator, is in charge of coordinating the multiple requests for load-balancing from different network links and preventing the conflicts among them.

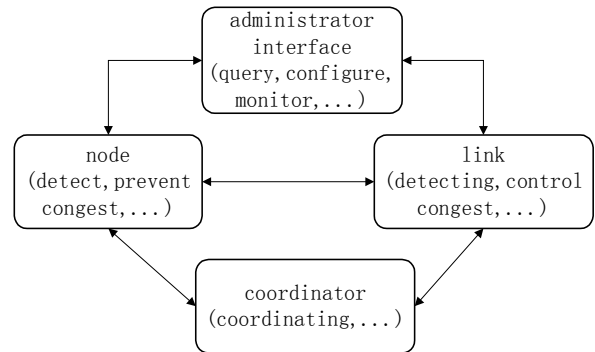


Figure 4 Role Model for network flows management prototype

## 4. SYSTEM DESIGN

The models involved in this stage are agent model, task model and information model. The former two models belong to the management part of the system, which are based on the previous analysis models; the last one belongs to the managed part, which is the abstract of network resources from requirements specification. The final product of this stage is the architecture for the target network management system, which is built on the basis of the three models. Architecture provides an effective way to combine these models together.

### 4.1. Task Modeling

Task model describes the interactions among multiple agents, or the behavior of roles. A task is defined as a group of sequenced activities for the purpose of realizing a certain goal in a given context. A task has the following primary features<sup>[7]</sup>: 1) one definite start and several definite ends; 2) a finite set of activities that are independent on each other; 3) should be defined for a specific role, and can not be shared by two or more different roles; 4) interactions may exist between different tasks.

After Role Model is created, tasks associated with each role will be illustrated. Every goal involved in a role can have a task that details how the goal is accomplished. A task is specified graphically using a finite state automation. In object-oriented field, state chart is often used to describe the behavior of object in lifecycle. The most notable diagrams are OMT state chart and its extension, UML state chart. Herein the UML state chart is borrowed to specify the tasks, named task diagram.

Figure 5 is a typical task diagram, which describes part of the load-balancing task that resides in the role of link initiator. In the multi-agent based network management paradigm, management applications are always realized by multiple agents cooperatively and coordinately. Therefore a management application is commonly divided to several parts that are distributed into different agents. When the system is running, the tasks in different agents would set up the communication path among them and work together in a cooperative manner to accomplish the system level tasks. In some sense, the task

model defines not only the behaviors of different agents, but also the rules that direct how these agents to behave and affect the whole system. Figure 5 only describes the initiator part of load balancing task, and the responder part is not included.

### 4.2. Agent Modeling

A group of agents are the primary building blocks to construct a multi-agent based network management system. Generally, an agent is always associated with some predefined functions and has to fulfill some responsibilities assigned to it [6][8]. Since a responsibility is generally mapped into one role, an agent often plays different roles simultaneously. The key to define agents is to make clear the roles that are required for each agent. This process is also called agent modeling.

Similar to the object model appeared in Object-Oriented field [9], the agent model here is used to describe the agents that constitute the distributed network management system. An agent model consists of two parts: agent classes and the relationships among them. An agent class is a template for multiple agent instances classified into the same category. At the same time, relationships between different agent classes are described as conversations occurred between two agent instances for realizing some management functions. An agent instance can not only change its roles dynamically in runtime, but also join two or more conversations simultaneously. Tasks and conversations are both the reflection of an agent's behavior. The difference is: conversation represents the outside behavior, while task is the description for the inner side of behavior.

Agent model in this paper is described by agent class diagram. Agent class diagram is very similar to object class diagram, but they have different semantics for the relationships between classes. Relationships in object-oriented have three types: association, inheritance and aggregation. But in agent class diagram, the relationship describes the conversation between different agent classes. An example for agent class diagram is shown as Figure 6. This diagram is drawn according to the previous analysis result of network flows management. Each block in the figure is an agent class, while the line with an arrow at the end denotes the conversation relationship between agent instances. Role names are included in each agent class, which point out what roles the agent can act as during execution. The direction of arrows is from the initiator of the conversation to the responder.

Identifying agent classes is the first step to construct agent model. Similar to the mapping from goals to roles, the mapping from roles to agent classes is also one to one. Considering an agent may play several roles simultaneously in runtime, so there also exists the case for multiple roles to map to a single agent class.

The second step is to identify the conversations between agent

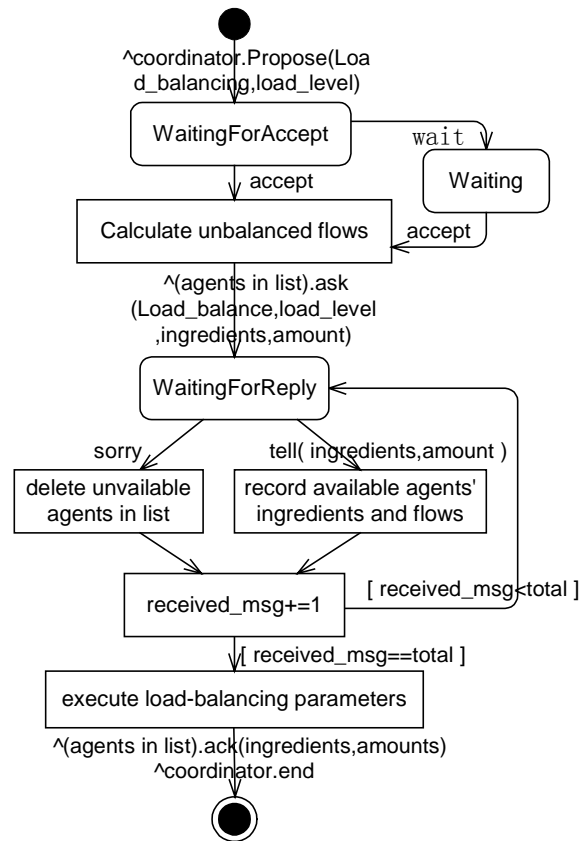


Figure 5 Task Diagram for Load-balancing inside the initiator role

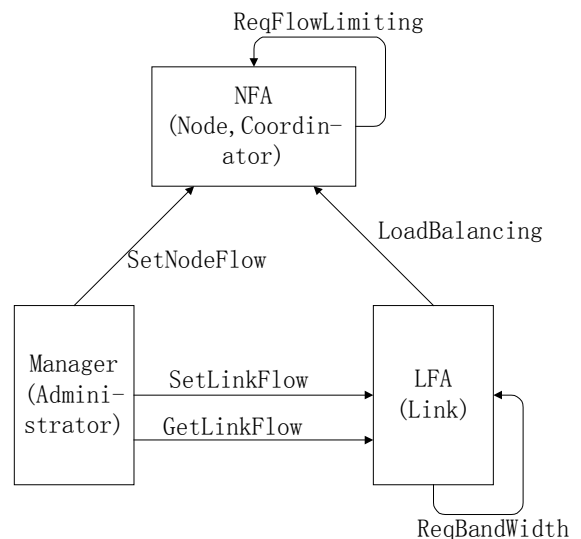


Figure 6 Agent Model for Network Flows Management

classes. As previously stated, in the role model, roles are connected through communication paths. Since roles will finally be mapped into agent classes, these communication paths can be directly mapped into conversations between agent classes. Therefore, the conversations' identifying tightly depends on the role identifying.

The last step is to draw the agent class diagram. This is an iterative process and the more detailed description of agent classes will be added with the advancement of development. In a word, the objective of agent modeling is to identify all the agent class for locating all the communication relationships.

### 4.3. Information Modeling

Information model plays an important role in traditional network management systems. The object-oriented technology has been widely adopted in current network management field to organize management information, which can effectively improve the compatibility, inoperability and scalability of the management system. In order to provide a better compatibility with other legacy systems, the information model is also kept as a part of the modeling approach of this paper.

The core concept for information model is managed objects (MO). Managed objects are the abstracted descriptions for network resources after filtering out some detail information. Managed objects encapsulate inside the differences of the network resources that belong to the same type so that from outside the same management features can only be seen. In this way, the complexity for integrating the management system with heterogeneous network devices is largely decreased. Similar to object model, information model also provides a way to organize managed objects and their relationships.

As a special kind of object model for network management field, information model can also be built by using object-oriented technology. ISO and ITU developed GDMO (Guidelines for The Definition of Managed Objects)<sup>[10]</sup> to describe the managed objects in a formalized way. Although GDMO has been widely employed in telecommunications world, it has several very negative shortcomings, such as complex notation syntaxes, hard to understand, etc. Therefore, this paper adopts the OMT (Object Modeling Technology) to construct information model. In theory, these two methods are equivalent to each other. Figure 7 is an example for information mode that was designed for a local telephone network. This figure contains two sub-diagrams: (a) is for the inheritance relationships between classes; (b) is for the aggregation relationships. The blocks represent different managed object classes, while the lines with various little symbols at the end denote the relationships between objects. To be more clearly, the blocks drawn in solid lines are the standardized objects from some

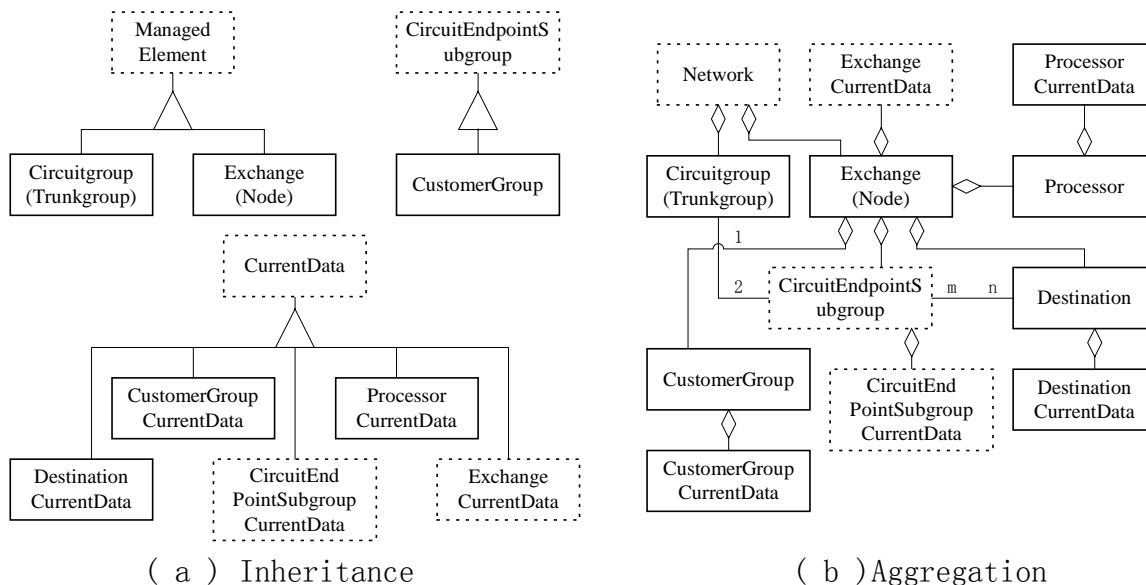


Figure 7 The Information Model for A Local Telephone Network

international or national standards, and those in dotted line are designed to meet some peculiar management functions.

#### 4.4. Management System Architecture

The last step in the design phase is to design the architecture of management system. In a common sense, the software agent-based network management system is a two-layered structure (Figure 8) <sup>[11]</sup>: the upper layer is composed of multiple agents, which is in charge of managing all the network resources in a distributed way; the lower layer is the managed network, which contains a great number of network resources. The management layer can also be further divided into more sub-layers if needed. The combination of hierarchy and distribution brings great advantages of higher flexibility, security and scalability than the traditional centralized management systems.

In the prototype system, we simply adopted the two-layered structure as the framework for the following implementation phase. Network resources exist in a simulated managed network and communicate with the upper management layer through traditional SNMP to exchange all types of management information. The agents in the management layer take charge of managing the network separately and autonomously. They can also communicate and negotiate with each other to accomplish a larger management tasks cooperatively.

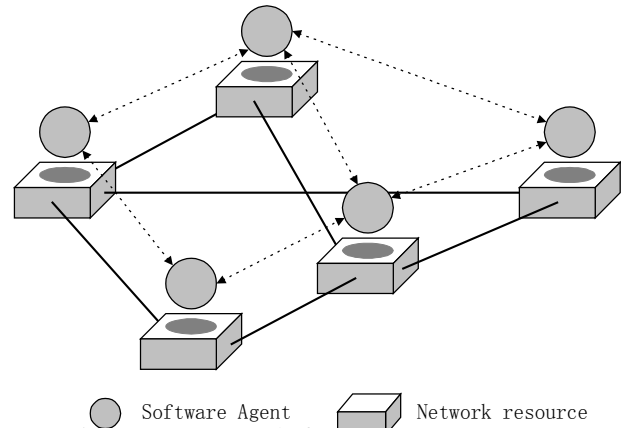


Figure 8 An Example for management System Architecture

### 5. A PROTOTYPE ON NETWORK FLOWS MANAGEMENT

The prototype is a multi-agent system (MAS) for managing multi-path network flows. Multi-path routing has been widely used in today's networks and it brings great flexibility and reliability to network system. However, it also brings tremendous difficulties to the management of various network flows that are composed of different types of sub-flows over all the network links. We developed a MAS system to schedule and control the Multi-path flows in a real-time model. Two types of agents are designed for network flows management: one is the Node Flow Agent (NFA), and the other is the Link Flow Agent (LFA) (as shown in Figure 6). The two kinds of agents always sit in the network switching nodes, named local nodes for the agents, such as exchanges, routers etc. The LFAs can communicate and negotiate with each other to reach a common understanding on the distribution of network flows among them and to balance their load through the flows' relocation. The NFAs are in charge of controlling the input flows according to the current situation of the flows in their local nodes. If some nodes have no enough capacity (link capacity and node capacity) for the input flows, the NFAs located in them will request the upstream LFAs for decreasing the related flow ingredients. The experimental result shows that the two agents can largely increase the usage rate of network resources, and prevent the network from congestion effectively.

In the implementation of the prototype, Java-based Agent Framework for Multi-Agent Systems (JAFMAS) is chosen as the platform to build management system. JAFMAS is a generic agent development platform finished by Chauhan <sup>[11]</sup>, which provides a general framework and a great number of reusable facilities to construct multi-agent systems.

### 6. CONCLUSION

Managing network by software agent has attracted great attentions as an emerging technology, and a great number of researchers accumulate in this field. White made a deep research on modeling network for management applications using mobile agent <sup>[12]</sup>, Cheikhrouhou proposed an intelligent agent architecture suitable for network management <sup>[13]</sup>, Hayzelden put forward a multi-agent approach for resource configuration in communication networks <sup>[14]</sup>. However, these researches are based on their own private architectures and development methods. So it is difficult to migrate the approach successfully used in one research to another. In other words, the lack of universality is a big problem faced with software agent-based network management systems.

From the viewpoint of modeling, we consider the system development as a process to build the model for the final target system. Therefore, we proposed a series of models to guide the system analysis and design stage by stage. These models provide a structured approach to describe features of the system from different standpoints. With the development furthering, the description will become more and more detail. This systematical modeling approach provides a more general way to build a distributed network management system by using software agent technology.

In order to testify the modeling approach, a prototype for managing network flows was developed. This experimental system realized some basic but very important management functions, such as load balancing, congestion control, etc. This simple experimental prototype runs on a simulated telephone network, and can manage the network flows effectively in an automatic and cooperative manner. It also provides a good foundation for the future application of the prototype result to a real network.

## REFERENCES

1. J.P. Martin-Flatin and S. Znaty. "A Simple Typology of Distributed Network Management Paradigms". Proc. 8th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'97), Sydney, Australia, October 1997: 13-24.
2. H. S. Nwana and D.T. Ndumu. An Introduction to Agent Technology. BT Technology Journal. Volume 14, Number 4, October 1996.
3. N R Jennings. A roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent System, 1998(1): 275~306
4. N. R. Jennings. Agent-Oriented Software Engineering. Proc. 12th Int Conf on Industrial and Engineering Applications of AI, Cairo, Egypt, 1999:4-10.
5. M. Wooldridge, N. R. Jennings and D. Kinny (1999) "A Methodology for Agent-Oriented Analysis and Design" Proc. 3rd Int Conference on Autonomous Agents (Agents-99) Seattle, WA: 69-76.
6. E. A. Kendall Goals and Roles: The Essentials of Object Oriented Business Process Modeling. the ECOOP Workshop on Object Oriented Business Process Modeling, June 1998
7. C. A. Iglesias, M. Garijo et al. Analysis and design of multiagent systems using MAS-CommonKADS. In AAI'97 Workshop on Agent Theories, Architectures and Languages, Providence, RI, July 1997.
8. M. F. Wood and S. A. DeLoach. An Overview of the Multiagent Systems Engineering Methodology. The First International Workshop on Agent-Oriented Software Engineering (AOSE-2000). Limerick, Ireland. June 10, 2000.
9. T. Quatrani. Visual Modeling with Rational Rose and UML. Addison Wesley, 1998.
10. ITU-T. X.722. Information Technology – Open Systems Interconnection – Structure Of Management Information: Guidelines For The Definition Of Managed Objects. 1992.
11. D. Chauhan. A Java-based Agent Framework for Multiagent Systems Development and Implementation. Ph.D. thesis, ECECS Department, University of Cincinnati, 1998.
12. T. White, B. Pagurek and A. Bieszczad. Network Modeling for Management Applications Using Intelligent Mobile Agents. Journal of Network and System Management, 1999, 7(3):295-321.
13. M. M. Cheikhrouhou. A software Agent Architecture for Network Management in Network Management: A State-of-the-Art. Networking and Information Systems, 1998, 1(1):9-38.
14. A.L.G. Hayzelden. A Multiple-Agent System Approach for Resource Configuration in Communication Networks. PhD Thesis (University of London),1999.

\* [zhangpeng@lucent.com](mailto:zhangpeng@lucent.com); phone 86-10-68748088 ext. 8569; fax 86-10-68748225; Bell Labs Research China, Lucent Technologies; Bell Labs Research China, Hui Long Technical Center, No.30 Hai Dian Nan Lu, Beijing 100080, P. R. China; \*\*[congzhao@263.net](mailto:congzhao@263.net); phone 86-29-2234565; Neocomputer Research Institute, Xi'an Jiaotong Univ., Xi'an 710049, P. R. China; \*\*\*[lzz@sun20.xjtu.edu.cn](mailto:lzz@sun20.xjtu.edu.cn); phone 86-29-2668642; Institute of Computer Architecture & Networks, Xi'an Jiaotong Univ., Xi'an 710049, P. R. China.