A Multilevel Policy-Based Network Management System for Differentiated Services Network

Yu Kang, Song Ouyang

Department of Computer Science, Central South University, Changsha, Hunan, P. R. China 410083

ouyangsong@yahoo.com

Abstract

Policy-based network management is a flexible, automatic and promising paradigm for network configuration management. The IETF has developed several specifications for policy-based configuration management. However, recent research seldom focus on a comprehensive solution for several core issues such as the rationality of the levels of the policies, the consistent relation between the policies and the architecture of the higher-level business management system, and the implementation of the multilevel policies. In this paper, a multilevel policy-based network management system for Differentiated ser-vices network is presented. The focus of the paper are the structure of the multi-level polices that is consistent with the architecture of the network management system, the policy storage and the policy processing that are independent of the individual devices, and Java implementation of the policy rules.

Keyword: Policy based network management, Differentiated services network, Multilevel polices, Policy server, QoS.

I. Introduction

The Internet today is in a fast transitional process. One of the most significant symbols of it is the best-effort service model where all transmissions are considered equal is converting to a new model that can provide predictable and different service levels for specific Quality of Service (QoS) requirements. To deliver QoS in IP networks, two approaches have been proposed: Integrated Services (IntServ) [1] uses the Re-source Reservation Protocol (RSVP) [2] to provide per-flow QoS supported by dynamically reserving resources on RSVP-enabled routers. Each flow is identified by the destination IP address. The scalability of the system with this approach is very poor as the routers must maintain a lot of information about the application flows and their reservations and must process a large number of messages for each reserved flow. Differentiated Services (DiffServ) [3] is a much simpler approach than IntServ. The QoS information is encoded in the Type of Service (ToS) byte in the IP header to identify different classes of service. Only edge routers in the DiffServ architecture need to perform the classification of traffic flows. The core routers only need to queue and schedule packets according to the value of the ToS field. This ensures that DiffServ has much better scalability characteristics and is therefore becoming more popular. On the other hand, this transition process makes the network management more difficult. It requires network management system should focus on the service and the traditional network management is not suitable anymore.

Policy-based network management (PBNM) has emerged as a promising paradigm for network operation and management [4,5]. Policies are rules governing the choices in behavior of a system [6]. It is a way to guide the behavior of a network through high-level declarative directives [7].

The initial work of the IETF was the definition of the Common Open Policy Ser-vice (COPS) [8] that is a protocol that allows the exchange of policy information between a Policy Decision Point (PDP) and a Policy Enforcement Point (PEP). PDP is defined as the point where policy decisions are made, whereas the PEP is the point where the policy decisions are actually enforced [9]. COPS is an adaptable protocol because it allows a number of different client types. These determine the structure and storage of the policy information to be exchanged. There are two major client types that are defined by the IETF: the Outsourcing mode (using the COPS-RSVP protocol [10]) and the Provisioning mode (using the COPS-PR protocol [11]).

The Policy Framework working group extended this work to establish a basic architecture for policy systems. They defined a PDP logical entity that makes policy decisions for itself or for other network elements that request such decisions [12]. The functional architecture was described in an (now expired) Internet Draft [13] as part of the policy framework. The new architecture added two roles: the Policy Repository (PR) and the Policy Management Tools (PMT). The Policy Repository is an entity that provides persistent storage. The Policy Management Tools enables an entity (e.g. person, application) to define and update policy rules and optionally, monitor their deployment [13]. This is the generally accepted architecture for PBNM systems.

The remainder of the paper is organized as follows: The policy rationality is analyzed in section 2, the system architectures are designed in Section 3, and the four levels of Diffserv policies are explained in Section 4. The policy database structure is explained in Section 5, and the collision detection is explained in Section 6.

II. Analysis of Policy Rationality

Policies can either be abstract or concrete [13]. Abstract policies specify a goal, objective or constraint that needs to be achieved without specifying how it is to be achieved. A concrete policy specifies a process or procedure that explicitly needs to be followed. Concrete policies are typically easy to achieve, because there are no details that are unclear or unspecified. Abstract policies have missing details, and thus the autonomous entity requires enough knowledge and intelligence to work out how it can be achieved, and how the outcome can be measured. Policies are also considered hierarchical, because a high-level abstract objective is decomposed into smaller and more specific goals until the policies become concrete and implementable. This process is referred to as policy refinement, translation or transformation in the literature. Since policies describe the behavior of a system; a model of that system is required to abstractly represent the real system that is under management. Increasing the level of abstraction of the model, increases the power of the specification and allows it to apply to more managed entities, but reduces the specific detail about how the goal is to be achieved or measured.

Comparing the network management levels, which service management level, net-work management level, network element management level and network element level, we divide policies into four different levels: service level policy (SLP), network level policy (NLP), device level policy (DLP) and instance level policy (ILP). We regard SLP, NLP and DLP as abstract policy because they only describe the management goal of each network management level. The last level policy, which is ILP, is regarded as a concrete policy. It not only describes the instance level management, but also explains how to use the MIB of SNMP or the PIB of COPS to implement the

management goals. Each level policy is connected to the neighbor level policy, be-cause the high-level policy has to be translated to low-level policy next to, policies have been translated to the configuration command the individual devices. Of course, each level policy is only for its corresponding network management level, which means SLP only concerns about the service management level requirements NLP only concerns about the network level management level requirements, etc. One of the main advantages of the four level policies is that when the requirements of specific network management level have been changed, only the corresponding level policy should be adjusted, and the other level policies should not be changed in most cases. It makes the policy model more efficient, scalable, and robust.

III. System Architecture

This section describes our general multilevel policy-based configuration management architecture shown in Figure 1. It contains four kinds of components: policy server, policy repository, PDP, and PEP.

The policy server is a component that deals with the abstract policies and is responsible for the traditional network management requirements, such as getting the topology information of the network, receiving the network events. Three major function modules are in policy server: basic information database, policy receiver and policy executor. Policy receiver gets service information from user interface, and then sends this information to policy executor. Policy executor stores the policy rule. By combining the user information in basic information database, it translates the service information into SLP. After the first translation, combining the network information in basic information database, policy executor will translate the SLP into NLP, and then from NLP into DLP. When each translation step has been finished, policy executor stores it into the policy repository. Basic information database has stored some information in a traditional way, including user information and network information, etc. User information includes user IP Address, SLA description, etc. Network information includes the network topology and devices information (in differentiated ser-vices network, devices information are routers IP Address and the configuration in-formation of these routers), etc, which are collected by the traditional network management module, such as configuration management module and fault management module. In this paper, we will not discuss these traditional management functions.

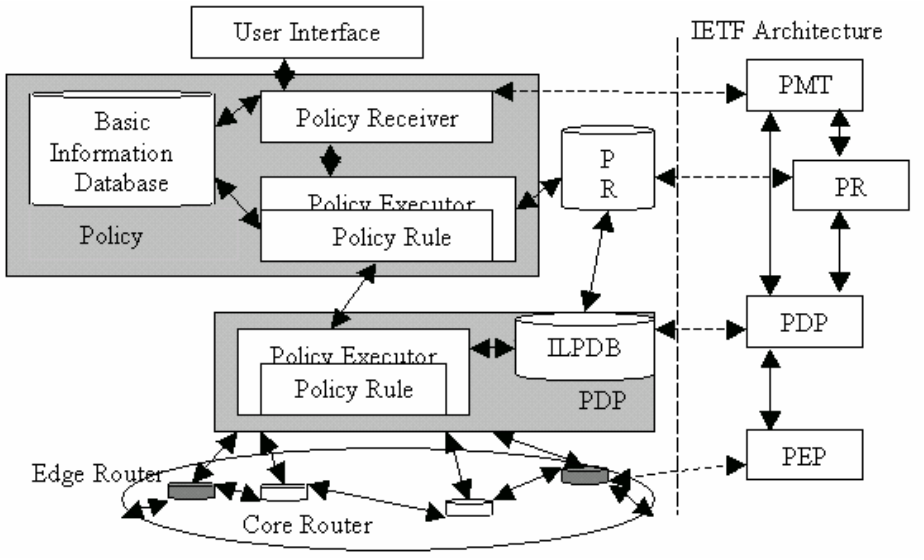

Fig. 1. Multilevel Policy-based Network Management System Architecture

The instance level policy database (ILPDB) has the same ability of the IETF standard policy repository. In our system, policy repository stores the abstract policy including SLP, NLP and DLP. The structure of ILPDB will be discussed in section 5.

In IETF standard architecture, PDP is the point where policy decisions are made. It performs the functions of retrieving and interpreting policies, detecting policy conflicts (we will discuss the detection of policy conflicts in section 6), receiving policy decision requests from PEPs, and returning policy decisions to them. Triggers to evaluate one or more policy rules can be events, polling, and explicit component requests. IETF policy framework does not include triggers explicitly. Only conditions (including timer conditions) are included. In our system, PDP translates the abstract policy (only DLP), which is stored in policy repository, into ILP, and then, PDP makes the decisions from the ILP. There are two major modules in PDP, which is Policy executor and instance level policy database (ILPDB). Policy executor in PDP has almost the same ability of one in policy server. The difference between these two executors is that policy executor in PDP only translates policy once, from DLP to ILP. After translation, policy executor will store the ILP in the instance level policy database. When the network events (In our system, these events include time events and network events, in section 4.) occur, policy executor in PDP will retrieve and interpret IDL from instance level policy database, and configuration commands of individual devices. Finally, PDP will use a network management proto-col, such as SNMP or COPS to send the configuration commands to the PEP. In our system, PDP is the point that communicates with the managed devices directly. So, PDP has to know the specific information of the individual devices.

Instance level policy database is the module that is used to store ILP and the individual devices information. The policy executor can use this information to make the decision. The structure of instance level policy database will be discussed in section 5.

In this paper, we divide policy into two types: abstract policy and concrete policy. The advantages are as follows: Firstly, PDP does not concern about the information of the network management level, such as network topology; Secondly, abstract policy can be used in any network environment, which adds the scalability of the system. Finally, when a lot of PEP are added into network, PDP does not need to change the policy rule, because the topology is concerned by the abstract policy, which reduces the burden of the PDP. These make model more efficient, scalable, and robust.

The PEP is the target entity that hosts the network elements where policy decisions are actually enforced. It is the target of a policy action when the rule condition evaluates to true. The separation of PEP and PDP is a logical, and not necessarily a physical separation. PEP and PDP may be combined and co-located.

IV. Designing and Implementation of Multilevel Policy Rule

Before discuss the design of the multilevel policy rule, two important concepts should be distinguished: policy rule and policy. Policy rule describes the framework of policies; The Policy Framework WG defines an aggregation of Policy Rules. Each policy rule comprises a set of conditions and a corresponding set of actions that are intended to be independent. Policy Rules are of the form: if <condition> then < action>. The <condition> expression may be a compound expression and it may be related to entities such as hosts, applications, protocols, users, etc. The <action> expression may be a set of actions that specify services to grant or deny. A policy is the instance of a policy rule. Comparing to the concept of the Object Oriented Program, The relation between policy rule and policy is like the one between class and object.

The policy server is a component that deals with the abstract policies and is responsible for the traditional network management assignments.

4.1 Java Network Policy Rule

In this paper, a multilevel policy rule model is proposed based on IETF standard policy rule model PCIM/PCIMe and is implemented in Java. This multilevel policy rule is referred as Java network policy rule (JNPR). The structure of JNPR is shown in figure 2: Because PCIM/PCIMe only describe the behavior of the policy rule, not mention how to implement them, the first step of JNPR is to construct the core rule which is consistent with PCIM/PCIMe. The core policy rule only implements the behavior of abstract policy. In specific network environment, such as differentiated services network, we should expand the function of JNPR core rule to meet the specific management requirements. The expansion of JNPR core rules use the concept of four level policies that is discussed in section 2. There are four different level policy rules in JNPR, which are service level policy rule (SLPR), network level policy rule (NLPR), device level policy rule (DLPR) and instance level policy rule (ILPR).

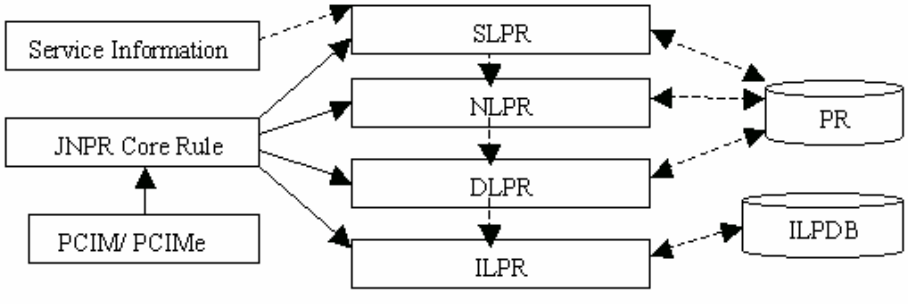

Fig. 2. Java Network Policy Rule

JNPR is used to produce policies. The procedures are: when service management information is received, it will be translated into service level policy (SLP), network level policy (NLP), device level policy (DLP) and instance level policy (ILP) step by step. Then SLP, NLP and DLP will be stored into PR and ILP will be stored into instance level policy database.

4.2 Implementation

4.2.1 basicPolicy Package

The basicPolicy package contains the classes and interfaces of Policy, Condition, Action and Event: Policy represents a basic abstract policy. A derived class of Policy represents the class of different level policies, which is executed when the Policy is started. PolicyRule registers instances of implementations of the Condition and Action. Two general Condition classes are provided by basicPolicy package: Simple-Condition, consists of single variable and value; can be used by PolicyRule that shall be executed based on a triggering event without further conditions; CompositeCondi-tion is a condition that consists of one or more SimpleCondition. SpecificCondition and Action classes can be derived from a Policy Class. The evaluations of PolicyRule are always initiated by Events, which are triggered by EventReceiver. Timer is a time based event receiver that generates TimeEvents at specific points in time. The abstract class Element is the parent of all network elements modeled by domain specific pack-ages. Roles can be assigned to Elements so that they can be used in conditions to select subsets of elements that should be affected by a PolicyRule.
PolicyRule.

4.2.2 diffServ Package

The specific domain targeted in this paper is the configuration of differentiated ser-vices nodes. Hence, the diffServ package contains element classes to represent differentiated services elements, such as conditions, meters, actions, droppers, queues, schedulers, etc. The methods of these classes allow the policy developer to create, delete and modify the elements. The classes have been modeled based on the DIFFSERV-MIB data model, i.e., these data path elements can be plugged together (with certain limitations) through methods provided by the common parent class DiffServElement, which is a child class of the Element class. The package defines class using inheritance, e.g. Marker and Dropper are derived from DiffServElement. MFFilter is derived from Filter that is also derived from DiffServElement.

4.2.3 multilevelPolicy Package

To expand functionalities of the basic abstract policy, we define ServiceLevelPolicy, NetworkLevelPolicy, DeviceLevelPolicy and InstanceLevelPolicy in multilevel policy package, which are all derived from Policy class.

4.2.4 Example

We use a service example to explain the structure of JNPR. This example contains information of event, conditions and actions:

User A has access to the high quality Video from
2005-1-1 to 2005-6-20;
Time is from 6:00 AM to 22:00 PM;
The SLA description of the high quality Video is AF;

In our system, Policy server will receive this abstract service information from user interface. To understand this information, policy server will use policy executor and PR to translate and store it. The service information will be translated into SLP, NLP, DLP and ILP. In our system, the form of these policies is described by JNPR. The form of SLP, NLP, DLP and ILP in JNPR is consisted of event, condition and action. The SLP is as follow:

if (Jack.ServiceType==HighQualVoD && Date>=2005-1-10 && Date<=2005-6-20 && Time>6:00 && Time<22:00)
{ HighQualVoD.priority=AssuredForwarding.priority}

From SLP to NLP, the policy will be added the network information. In this paper, we concern about differentiated services network environment. NLP is as follow:

If (Date>=2005-1-10 && Date<=2005-6-20 &&
Time>6:00 && Time<22:00 &&
User_A.IPAddress==192.168.0.1 &&
HighQualVoD.Port==1024 &&
HighQualVoD.DiffServCodePoint==AssuredForwarding11 &&
AssuredForwarding11.DSCP==001010b &&
AssuredForwarding11.Scheduling==PriorityQueuing)
{ Configure.EdgeRouters=[192.168.1.1,192.168.4.1]；
Configure.CoreRouters=[192.168.2.1,192.168.3.1]； }

After the NLP has been set up, NLP would be translated into DLP. The process of the translation from NLP to DLP adds different type device's information into DLP. The form of DLP is as follow:

If (Router == EdgeRouters) {
If (Date>=2005-1-10 && Date<=2005-6-20 &&
Time>6:00 && Time<22:00&&

UDPPacket.DestIPAddress==192.168.0.1 &&
UDPPacker.Port=1024)
{UDPPacket.DSCP=AssuredForwarding11.DSCP； } }
If (Router == CoreRouters) {
If ((Date>=2005-1-10 && Date<=2005-6-20 &&
Time>6:00 && Time<22:00&&
IPPacket.DSCP==AssuredForwarding11.DSCP)
{(Queue1.Enqueue(IPPacket))}

After these three level abstract policies have been formed, policy server will send the DLP to PDP. PDP will translate DLP into ILP. Because ILP is a concrete policy, it connects to specific protocol and individual devices. In this paper, we give the example of ILP with SNMP operation. The form of ILP is as follow:

If (Router == 192.168.2.1&& ManageProtocol==snmp) {
If(Date>=2005-1-10 && Date<=2005-6-20 &&Time==6:00 &&
UDPPacket.DestIPAddress=192.168.0.1 &&
UDPPacker.Port=1024)
{SNMP.Set(1.3.6.1……., XXXX); ……
SNMP.Set(1.3.6.1……., XXXX);}}
If (Router == 192.168.2.1&& ManageProtocol==snmp) {
If (Date>=2005-1-10 && Date<=2005-6-20 &&Time==22:00 &&
UDPPacket.DestIPAddress=192.168.0.1 &&
UDPPacker.Port=1024)
{SNMP.Set(1.3.6.1……., Null);……
SNMP.Set(1.3.6.1……., Null);}}

The Action in ILP is the concrete management operation. In this paper, we use SNMP set operation to explain the structure of action. The SNMP set operation contains is connected with several differentiated services MIB tables [14]. In this paper, the SNMP set operation, which is replaced by "XXXX ", is that PDP sends action parameters of ILP to Policy Management MIB and to a domain specific MIB module of specific managed devices, which supports SNMP; For the policy-rule example considered before policyFilter is going to be: DestIPAddress is 192.168.0.1 and sourceL4Port is 80 and rate is below 100kB/sec; PolicyAction is: AF service. The PEP (or the managed device) informs the PDP about its capabilities; For this purpose the SNMP message should be used as it is an acknowledged one, thus the agent knows that a manager is aware of its capabilities. This is especially useful in case of multiple managers. The DiffServ policy module set the appropriate values either directly or via DiffServ MIB module. In the example, the DiffServ Policy module will populate the DiffServ module tables to implement AF for User_A on the ingress interfaces specified by the roles in the pmRoleESTable.

V. Policy Database Design

In this section we will discuss the structure of instance level policy database, which is used to handle the policies. The ILPDB is defined according to the IETF specifications. A brief overview is given in the following paragraph.

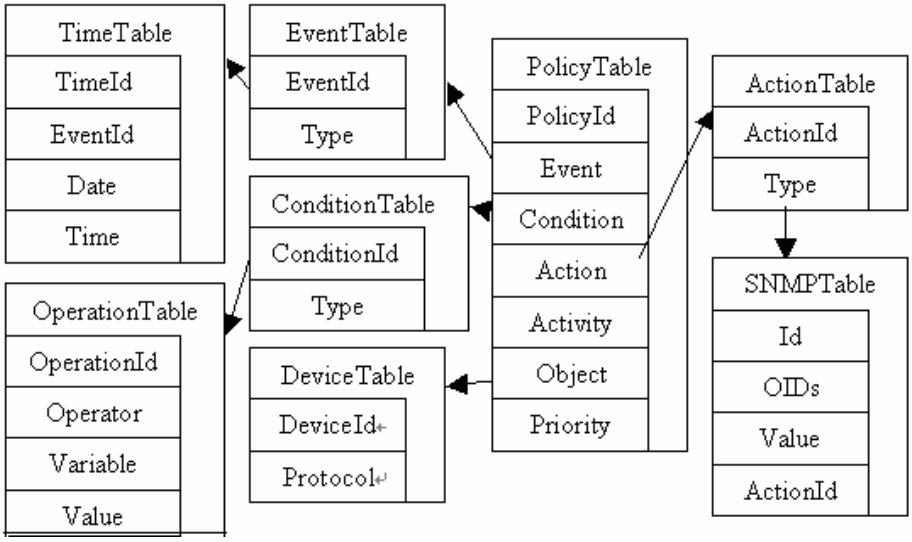

Fig. 3. Instance level policy Database

The ILPDB is divided into five groups (see Fig. 3):

- The PolicyTable contains the items that form the policies, define their Priority item in case of conflicts, and use the Activity item report their status.
- The ConditionTable provides items for forming the conditions of the policies.
- The ActionTable includes the items that define the actions of the policies.
- The EventTable includes the items that define the events of the policies.
- The other tables are served for the tables mentioned above.

The policy condition and actions must be installed in the appropriate tables of the Condition and Actions: The condition is placed on the ConditionTable. Protocol item in DeviceTable determines the action type in policy. The SNMPTable is used to map the action type to the policy action to the SNMP.

VI. Conclusions and Future Work

The work introduced here is a part of efforts that try to deliver an integrated solution for some key issues in the policy-based network management system. The multilevel policy structure and the JNPR can meet different service requirements. Network rules specified within our framework are dynamically triggered by events. This dynamic configuration of policy forms the basis of the adaptive management.

As a future work, we intend to further enhance the multilevel policy: to investigate whether the hierarchies of policies can reduce the burden of PDP, and how the pro-posed instance level policy database should be modified (if necessary).

References

[1] 1. Braden, R., Clark, D. & Shenker, D.: Integrated Services in the Internet Architec-ture: an Overview, RFC 1633, (June 1994)

[2] 2. Braden, R., Zhang, L., Berson, S., Herzog, S. & Jamin, S.: ReSerVation Protocol (RSVP) Version 1 Functional Specification, RFC 2205, (September 1997)

[3] 3. Bernet, Y., Smith, A., Blake, S. & Grossman, D.: An Informal Management Model for DiffServ Routers, Internet Draft, draft-ietf-diffserv-model-06.txt, (Feb. 2001)

[4] 4. S. J. Shepard: Policy-based Networks: Hype and Hope, IT Prof., vol. 2, no. 1, (Jan.-Feb. 2000) 12–16

[5] 5. Introduction to Policy-Based Networking and Quality of Service, IPHighway, White Paper, (Jan. 2000)
[6] 6. M. Sloman and E. Lupu: Security and Management Policy Specification, IEEE Network, (2002)10-19
[7] 7. P. Flegkas, P. Trimintzios, G. Pavlou, I. Andrikopoulos, and C. F. Cavalcanti: Policy-based Extensible Hierarchical Network Management, presented at Proceed-ings of Workshop on Policies for Distributed Systems and Networks (Policy 2001), Bristol, U.K, (2001)
[8] 8. RFC 2748, D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry: The COPS (Common Open Policy Service) Protocol, Network Working Group, (January 2000)
[9] 9. RFC 2753, R. Yavatkar, D. Pendarakis, and R. Guerin: A Framework for Policy-based Admission Control, Resource Allocation Protocol, (January 2000)
[10] 10. RFC 2749, S. Herzog, J. Boyle, R. Cohen, D. Durham, R. Rajan, and A. Sastry: COPS usage for RSVP, Network Working Group, (January 2000)
[11] 11. RFC 3159, K. McCloghrie, M. Fine, J. Seligson, K. Chan, S. Hahn, R. Sahita, A. Smith, and F. Reichmeyer: Structure of Policy Provisioning Information (SPPI), Resource Allocation Protocol, (August 2001)
[12] 12. RFC 3198, A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser: Terminology for Pol-icy-Based Management, Policy framework, (November 2001)
[13] 13. draft-ietf-policy-framework-00.txt, M. Stevens, W. Weiss, H. Mahon, B. Moore, J. Strassner, G. Waters, A. Westerinen, and J. Wheeler: Policy Framework, Policy Framework, 13 (Sept 1999).
[14] 14. Fred Baker et al.: Management Information Base for the Differentiated Services Architecture, Internet Draft, draft-ietf-diffserv-mib-02, (March 2000).

Yu Kang, He received his bachelor and master degree from Central South University, Changsa, Hunan, China. His research interest is network management system.

Song Ouyang, is Professor at the Central South University, Changsha, Hunan, China. He received his Master of Science degree from, and his Ph.D from University Hertfordshire, UK. His research interests are Distributed System, Software Engineering.