A Conceptual Model of Service Customization and Its Implementation

Su-Bin Shen[1], Guan-Qun Gu[2], and Shun-Yi Zhang[1]

[1]*Research Center of Network Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003 P.R. China*

[2]*Department of Computer Science and Engineering, Southeast University, Nanjing 210096, P.R. China*

E-mail: sbshen@njupt.edu.cn; ggu@seu.edu.cn; dirzsy@njupt.edu.cn

Received April 27, 2002; revised January 13, 2003.

Abstract With the development of Internet and next generation networks in telecommunications, more and more new services are required to be introduced into networks. Introducing new services into traditional network is always associated with standardizing new protocols. The progress of protocol standardization usually takes several years, which cannot meet the increasing demands of the applications in Internet and next generation networks. Service customization in network systems may be one possible solution to cope with this problem. Based on the principle that network service is provided by interactions among protocol entities, this paper proposes a conceptual model of service customization (SECUM) by separating the service logic from protocol interactive logic within existing network architecture. The theory of Communicating Sequential Processes (CSP) is used to formalize the SECUM in order to locate exactly the service logic and to define precisely the SECUM. For validating the SECUM's usability in practical network systems, this paper also proposes an implementation model for SECUM: a component-based protocol implementation model (CPIM). CPIM discomposes protocol entity into application component, service component, message component and communication component. Service component associates application component with message component. Users or network managers can customize network services by configuring service component. The paper shows respectively the applications of SECUM and CPIM by proposing a customizable IP service model based on SECUM and describing an implementation of Session Initiation Protocol (SIP) based on CPIM. Compared with the existing service-customization techniques, SECUM is a service customization model internal to network system and may provide more powerful capabilities of service customization.

Keywords service customization, protocol implementation, network service, CSP, IP service, SIP

1 Introduction

The growing of Internet applications stimulates the innovations of Internet technologies. New technologies, on the other hand, put forward more requirements on Internet. For example, applications in multimedia communication have speeded up the research on multimedia networking technologies, such as service model (InterServ and DiffServ model), protocols (RTP, SIP, etc.) and QoS control mechanism (traffic engineering, QoS routing, etc.). The development in technology of multimedia networking gives rise to a series of novel applications in Internet, such as Internet telephony and multimedia conference. Internet telephony influences greatly on both Internet and telecommunication networks, so that development of next generation networks is proposed both on Internet and on telecommunication networks. The mutual promotion of application requirement and technical innovation in Internet shows, on one hand, that there are more and more requirements for developing Internet technologies. On the other hand, it shows that Internet technologies are not developed perfectly to satisfy the constantly changing requirements.

In order to meet this challenge, a series of active networking technologies are proposed in the 90s of the last century. Active Networks, Programmable Networks and Mobile Agents are some typical active networking technologies. They try to provide users/applications with the capabilities of service customization so that users/applications are able to design and modify network service according to their specific requirements.

*Regular Paper

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 69873008, 90104009, and 60273091, the National High-Tech Development 863 Program of China under Grant Nos. 863-306-QN2000-4 and 2002AA121069, by the Natural Science Funds of Jiangsu Province, China, under Grant No. BK2001205, and by Natural Science Funds of Universities of Jiangsu Province, China, under Grant No. 00KJB520002.

Active Network[1−3] proposes an approach to customize network service by introducing an active packet carrying programming code which can be executed in network intermediate nodes to customize these nodes' behaviors. This mode of service customization destroys the end-to-end transparency for IP packet transmission, increases overhead of packet processing in network nodes. It is not consistent with the current Internet goals of high scalability and high performance. So the future of the applications of Active Networks in Internet is not very clear now. Since Active Network proposes a way of service customization by transferring code within message, it may find some valuable applications in higher layers of network system. In fact, many researchers in the field of telecommunications are now showing great interests in Active Networks and have got some significant results in the applications of Active Networks to network management[4].

Programmable Networks[5,6] try to establish a virtual and open programming interface for network services to realize service customization above the existing physical networks, such as IP packet switching network and Public Switched Telephone Network (PSTN) based on SS7 signaling. The architecture of programmable Networks is compatible with that of the existing networks, so the idea of service customization in this mode is widely accepted in industry. The JAIN[7,8] based on Java technology and Parlay[9] proposed by Parlay organization are two typical open service platform technologies evolved from Programmable Networks. In our opinions, this mode of service customization will find wide applications in opening telecommunications network for third-party service provider to bring competition in providing telecommunication services.

Mobile agents evolve from Intelligent Agents in the field of Artificial Intelligence. Researchers in the field of computer network often intend to take the mobile agent as the agent that is the same in the network management with some mobile capability[10]. In our opinion, mobile agents will find practical applications in both service customization and network management. The most significant advantage of the mobile agent is to replace the client-server interaction over network with local interaction between an agent and a server so as to reduce the usage of network bandwidth and shorten the response time between request and reply.

According to the current network architecture[11,12], network service is the capacities fulfilled though protocol message interacts in network environment. Service is related with protocol in a network protocol entity[11]. From this principle, we can deduce an assertion that capacities of service customization are related with capacities of protocol customization. In order to provide a flexible and customizable network service, a flexible and customizable protocol entity should be provided. If the protocol entity is still take the form of monolithic structure, service customization will be limited only within the way of selecting options and parameters of message specified by protocol. In more than two years, we have been trying to validate this assertion. This paper will describe some results we have achieved in this research.

In our research work, we have found that most of current service customization technologies pay little attention to the support of underlying protocols. Programmable Network proposes an idea of establishing a virtual network to mask the physical network with programmable interface. This virtual network is only software abstraction of physical resource in the network. It is not clear how to relate the software abstraction interface, which is also called Application Program Interface (API), with underlying protocols. Active Networks and Mobile Agents only concern transferring and executing mobile codes, they do not concern the protocol model, the infrastructure supporting for execution of mobile code[13]. Associating the service customization with protocol composition is one of key points in our research on service customization.

In Section 2, we firstly describe an abstract service customization model, SECUM, which is used to specify service provision logic and service customization mode within one protocol entity. In Section 3, we describe a component-based protocol implementation model, CPIM, which is used to validate the availability of implementing SECUM. In Section 4, we introduce some work in customizing IP service and implementing SIP based on SECUM and CPIM respectively. In Section 5, we compare some related studies, and in Section 6, we give a conclusion of this paper.

2 Service Customization Model

When we talk about service customization, we should firstly clarify what it means by service customization. In our definition, service customization refers to capacities of providing service in some procedure or logic specified by network users or applications, not in the logic specified only by standardized protocol. According to our definition, there is no capability of service customization within the services provided by traditional protocol implementation. Its service logic has been specified by the standardized protocol and users are only able to select the options of standardized logic by choosing the values of parameters specified in the protocol.

2.1 Analysis of Traditional Network Service

Why is there no capability of service customization within the traditional network service? If we look into the implementation model of traditional protocol entity, we can find that service primitives and protocol messages are processed in the same state machine[14]. This means that service logic is combined with protocol logic to form a standardized protocol specification. So the service logic is also the standardized logic specified in protocol and users or applications have no right to customize the services. They can only select different logic branches by choosing different values of control parameters.

In our opinion, single state machine is the main factor for the shortage of service customization capabilities. In order to enhance the capabilities of service customization, we should separate the service logic from protocol logic in one protocol entity.

2.2 Assumptions of Service Customization Model

It can be concluded based on the analysis in last section that the logic of providing service to user within one protocol entity should be separated from the logic of protocol message interaction in order to enhance network protocol with service customization abilities. To clarify the scope of the service customization model to be defined, some assumptions of the model are made as follows.

Assumption 1. *Service customization model does not change the existing network architecture.*

We hope that the service customization model, SECUM, should be applied directly in existing networks. So it requires that SECUM should be compatible with the existing network architecture and suitable for hierarchical network structure.

Assumption 2. *Service customization model is limited within a single protocol layer.*

To be suitable for hierarchical network structure, the scope of SECUM should be limited to a single protocol layer, and provide the abilities of service customization within one protocol layer. Service customization model within a single protocol layer should provide service interfaces to upper layer, communication interfaces to lower layer, message interaction interfaces to peer layer and management interfaces to management layer.

Assumption 3. *Service customization model should be usable in all protocol layers.*

The networks in future should open network service not only in application layer, but also in several different layers. For example, they will open end-to-end singling protocol layer, or open directly the communication network layer composed by routers and switches. So, the service customization model should be usable in all protocol layers, that is, an abstracted service customization model for all protocol layers should be designed.

According to Assumption 1, there is no need to define another network architecture suitable for service customization. According to Assumption 2, the paper will focus on service customization model only within a single protocol layer. According to Assumption 3, service customization model can be used not only in service customization of application layer, but also in other layers, such as service customization of IP layer.

2.3 Definition of SECUM

We assume that our research on the service customization technologies is based on the current network architecture. The network architecture widely used now is of layered structure in which network functions are divided into layers. Within a layer, a set of services is provided by protocol entity exchanging messages with remote entities in the same layer. According to this architecture, customizing service of the whole network system can be transformed into customizing service of uppermost layer in that system. So the following part of the paper will focus on one layer service customization.

As we know, services in one layer are provided by protocol entity exchanging message with remote entities in the same layer. Services are specified by service primitives, which are associated with protocol messages within protocol entity. So we propose a new protocol entity model in which service logic and protocol logic are separated. This model can be used to specify the service customization, so we call it Service Customization Model, that is, SECUM (see Fig.1).

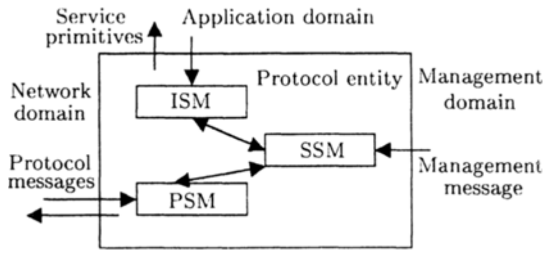

Fig.1. SECUM for end system entity.

We abstract the logic of processing message interaction between local entity and remote entities into a Protocol State Machine, that is, PSM. We abstract the logic of processing service primitive interaction between local entity and its users into an Interface State Machine, that is, ISM. So we can find that there is implicit state machine between PSM and ISM. This state machine can transform the user's request commands (service primitives) into a procedure of commands to trigger the processing of PSM messages. This procedure is, in fact, the service logic provided to the entity users. In our opinion, this state machine is just an abstraction of service logic. So we call this state machine Service State Machine, that is, SSM.

Fig.1 gives a form of SECUM in an end system node. For only in the end system of network node, the upper-most protocol layer should provide service to its users. In the intermediate node (such as IP layer in IP router), the entity of the upper most layer need not deliver the service to its upper users, but only needs to re-send (forward) protocol message to the network domain. So in the intermediate system, SECUM can be simplified by deleting ISM (see Fig.2).

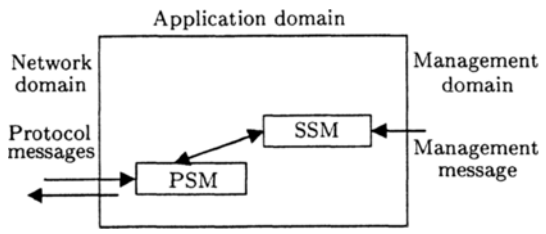

Fig.2. SECUM for intermediate system entity.

SECUM can be used to analyze Intelligent Network (IN), the current telecommunication service system. When a call request is received by switch (corresponding to PSM in SECUM), the special call number (such as 800, 300, etc.) in request triggers the switch to transfer the call request up to service server (Service Control Point of IN, corresponding to SSM in SECUM). Service server returns a series of commands according to the service logic to instruct the switch's actions for fulfilling the required service. The mode of service provision can be described with SECUM for the intermediate system. The only difference is that the switches and SCPs of IN are in different physical network nodes. But their logical interactions are the same. The simple example shows that SECUM is consistent with the service provision mechanism of practical network systems. It should be noted that IN can now only customize its service through management domain. There is also no capacity of service customization in it according to our definition.

2.4 SECUM Formal Model

In order to analyze the capacities of service customization with SECUM, a formal model of SECUM should be given. SECUM is a model that involves several state machines interconnected together by message interaction. We are concerned not only about messages exchanged among these state machines within one SECUM, but also about the messages exchanged between the SECUM and environments. The environments of SECUM include application domain, network domain and management domain as illustrated in Fig.1. In our point of view, we study the system logic only through the observable system behaviors. We try to specify the logic within the SECUM by describing the message interaction among the state machines of the SECUM.

We choose the theory of Communicating Sequential Processes (CSP)[15,16] as formal method to formally specify the SECUM. CSP is a concurrency theory, which is suitable for describing interactions among concurrent computing processes. According to our experience, the key points to use CSP are as follows: 1) find each sequential computing process from the system; 2) identify interactive channels among sequential processes; 3) define types of interactive messages among the sequential processes; 4) describe separately the behavior of each sequential process with interactive messages and channels; 5) compose all sequential processes with CSP composition operators to form the concurrent computing system model.

According to the requirements of modeling systems with CSP[15], we define application domain interacting with SECUM as application environment. ISM exchanges service primitives with its user through a pair of single directional channels C_{IA} (from ISM to application environment) and

C_{AI} (from application environment to ISM). We define network domain interacting with SECUM as network environment. PSM exchanges protocol messages with remote protocol entities through channels C_{PN} (from PSM to network environment) and C_{NP} (from network environment to PSM). We define management domain interacting with SECUM as management environment. SSM exchanges management messages with service manager through channels C_{SM} (from SSM to management environment) and C_{MS} (from management environment to SSM).

Inside SECUM, ISM exchanges service request and reply messages with SSM through a pair of single directional channels C_{IS} (from ISM to SSM) and C_{SI} (from SSM to ISM). PSM exchanges service request and reply messages with SSM through channels C_{PS} (from PSM to SSM) and C_{SP} (from SSM to PSM). Formal structure of SECUM for end system entity is illustrated in Fig.3.

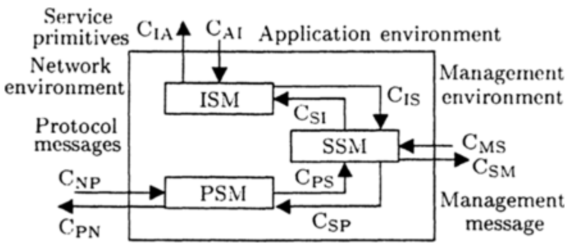

Fig.3. Formal SECUM for end system entity.

The sequential processes in which we are interested in ISM are as follows: (I1) ISM receives service request from application environment, which is transformed by interface logic (ILOGIC) into internal service request primitive and sent to SSM; (I2) ISM receives internal service reply from SSM, which is transformed by ILOGIC into service reply and sent to application environment. Combination of (I1) and (I2) can fulfill the service request initiated from application environment. In addition, ISM should have capabilities to deliver the remote service request to application environment and get its conformance. These sequential processes are as follows: (I3) ISM receives internal service request from SSM, which is transformed by ILOGIC into service indication and sent to applications environment; (I4) ISM receives service conformance from application environment, which is transformed by ILOGIC into internal service reply and sent to SSM. The combination of (I3) and (I4) can fulfill the conformance to remote service request.

Composing above sequential processes can form an ISM behavior model described with CSP (see Fig.4). Here, C_{AI}?inf_req represents that ISM receives message inf_req from channel C_{AI}. ILOGIC(inf_req) means that ISM deals with inf_req using process ILOGIC. C_{IS}!inf_srv_req means that ISM sends message inf_srv_req through C_{IS}.

$$\boxed{\begin{array}{lr}
\text{ISM} = ((C_{AI}?\text{inf\_req} \rightarrow \text{ILOGIC(inf\_req)} & \\
\quad \rightarrow C_{IS}!\text{inf\_srv\_req} \rightarrow \text{ISM}) & \text{(I1)} \\
\quad |||(C_{SI}?\text{inf\_srv\_repl} \rightarrow \text{ILOGIC(inf\_srv\_repl))} & \\
\quad \rightarrow C_{IA}!\text{inf\_repl} \rightarrow \text{ISM})) & \text{(I2)} \\
\quad |||(C_{SI}?\text{srv\_inf\_req} \rightarrow \text{ILOGIC(srv\_inf\_req)} & \\
\quad \rightarrow C_{IA}!\text{inf\_ind} \rightarrow \text{ISM}) & \text{(I3)} \\
\quad |||(C_{AI}?\text{inf\_conf} \rightarrow \text{ILOGIC(inf\_conf)} & \\
\quad \rightarrow C_{IS}!\text{srv\_inf\_repl} \rightarrow \text{ISM})) & \text{(I4)}
\end{array}}$$

Fig.4

The collection of types of messages transferred along C_{AI} in ISM can be represented as αC_{AI}(ISM). If there is no confusion, it can be simplified as αC_{AI}. In CSP model of ISM illustrated in Fig.4, αC_{AI} = {inf_req, inf_conf}, αC_{IA} = {inf_repl, inf_ind}, αC_{SI} = {inf_srv_repl, srv_inf_req}, αC_{IS} = {inf_srv_req, srv_inf_repl}.

In the theory of CSP, "a $\rightarrow$ P" means when event *a* occurs, it will behave the same as P. Here P is a process. "P;Q" means when process P is successfully completed, it will behave the same as Q. "P ||| Q" represents interleaving parallel composition between processes P and Q without synchronous constraints. It is noted here that "ILOGIC(inf_req) $\rightarrow$ C_{IS}!inf_srv_req" should be represented as " ILOGIC(inf_req); C_{IS}!inf_srv_req", for in the expression of "a $\rightarrow$ P", a cannot be a process according to the rules of CSP theory. In order to clarify the sequential processing relations, we simply abstract ILOGIC(inf_req) into an event and use the "$\rightarrow$" instead of ";" to express the relation between ILOGIC(inf_req) and C_{IS}!inf_srv_req.

The sequential processes in which we are interested in PSM are as follows: (P1) PSM receives message request from network environment, which is transformed by protocol logic (PLOGIC) into internal service request and sent to SSM; (P2) PSM receives reply from SSM, which is transformed by PLOGIC into message reply and sent to network environment; (P3) PSM receives service request from SSM, which is transformed by PLOGIC into message request and sent to network environment; (P4) PSM receives message reply from network environment, which is transformed by PLOGIC into service reply and sent to SSM.

These sequential processes are specified and composed formally with CSP to form a formal model of PSM (see Fig.5). In this formal model, $\alpha C_{NP} = \alpha C_{PN}$ = {msg_req, msg_repl}. It shows that interactions between PSM and network environment in two directions are equal. This means that they are in peer-to-peer relationship. αC_{PS} = {msg_srv_req, svr_msg_repl}, αC_{SP} = {msg_srv_repl, srv_msg_req}. It shows that interactions between PSM and SSM in two directions are not equal. This means that PSM and SSM are in a client-server relationship.

PSM = ((C_{NP}?msg_req $\rightarrow$ PLOGIC(msg_req)
  $\rightarrow$ C_{PS}!msg_srv_req $\rightarrow$ PSM) (P1)
  |||(C_{SP}?msg_srv_repl $\rightarrow$ PLOGIC(msg_srv_repl)
  $\rightarrow$ C_{PN}!msg_repl $\rightarrow$ PSM)) (P2)
  |||((C_{SP}?srv_msg_req $\rightarrow$ PLOGIC(srv_msg_req)
  $\rightarrow$ C_{PN}!msg_req $\rightarrow$ PSM) (P3)
  |||(C_{NP}?msg_repl $\rightarrow$ PLOGIC(msg_repl)
  $\rightarrow$ C_{PS}!srv_msg_repl $\rightarrow$ PSM) (P4)

Fig.5. CPS model of PSM.

The sequential processes in which we are interested in SSM are as follows: (S1) SSM receives interface service request from ISM, which is transformed by service logic (SLOGIC) into message service request and sent to PSM; (S2) SSM receives message service reply from PSM, which is transformed by SLOGIC into interface service reply and sent to ISM; (S3) SSM receives message service request from PSM, which is transformed by SLOGIC into interface service request and sent to ISM; (S4) SSM receives interface service reply from ISM, which is transformed by SLOGIC into message service reply and sent to PSM; (S5) SSM receives management request from management environment, which is handled by SLOGIC and SSM returns back management reply to management environment.

These sequential processes are specified and composed formally with CSP in Fig.6 to form a formal model of SSM. In this mode, αC_{MS} = {mgt_req}, αC_{SM} = {mgt_repl}, the definitions of αC_{IS}, αC_{SI}, αC_{PS} and αC_{SP} have been given in explanations to Fig.4 and Fig.5.

SSM = ((C_{IS}?inf_srv_req $\rightarrow$ SLOGIC(inf_srv_req)
  $\rightarrow$ C_{SP}!srv_msg_req $\rightarrow$ SSM) (S1)
  |||(C_{PS}?srv_msg_repl $\rightarrow$ SLOGIC(srv_msg_repl)
  $\rightarrow$ C_{SI}!inf_srv_repl $\rightarrow$ SSM)) (S2)
  |||((C_{PS}?msg_srv_req $\rightarrow$ SLOGIC(msg_srv_req)
  $\rightarrow$ C_{SI}!srv_inf_req $\rightarrow$ SSM) (S3)
  |||(C_{IS}?srv_inf_repl $\rightarrow$ SLOGIC(srv_inf_repl)
  $\rightarrow$ C_{SP}!msg_srv_repl $\rightarrow$ SSM)) (S4)
  |||(C_{MS}?mgt_req $\rightarrow$ SLOGIC(mgt_req)
  $\rightarrow$ C_{SM}!mgt_repl $\rightarrow$ SSM) (S5)

Fig.6. CPS model of SSM.

The SECUM behavior can be specified by ISM, PSM and SSM with synchronous parallel operator.

$$\text{SECOM} = \text{ISM}||\text{PSM}||\text{SSM}$$

Here "P || Q" represents that processes P and Q are composed with synchronous parallel operator "||", that is, if there is a channel C_{PQ} from P to Q, then event C_{PQ}!msg in P occurs if and only if C_{PQ}?msg in Q occurs.

2.5 Analysis of Service Customization

The sequential process (S5) in SSM describes mechanism of traditional service customization through management environment. If we make a simple extension to SECUM, we can describe with SECUM the mechanism of service customization used in Active Networks and Programmable Networks.

If we look at the SECUM illustrated in Fig.3, we can find that there are three ways to contact with SSM: by management environment, through ISM and through PSM. So there may be three ways to customize service: by management environment, through ISM and through PSM. In fact, service customization in Active Networks is through PSM, and in Programmable Networks it is through ISM. The traditional service customization is through management environment. In order to describe mechanism of service customization, we need to add two sequential processes in SSM: (S6) SSM receives a management request message, inf_mgt_req, from ISM, which is processed by SLOGIC and a management reply message, inf_mgt_repl, is returned to ISM. (S7) SSM receives management request message, msg_mgt_req, from PSM, which is processed by SLOGIC and a management reply message, msg_mgt_repl, is returned to PSM. In addition, αC_{IS} is expanded with "inf_mgt_req", αC_{SI} with "inf_mgt_repl", αC_{PS} with "msg_mgt_req", and αC_{SP} with "msg_mgt_repl". The formal descriptions of (S6) and (S7) are shown in Fig.7.

(C_{IS}?inf_mgt_req $\rightarrow$ SLOGIC(inf_mgt_req)
  $\rightarrow$ C_{SI}!inf_mgt_repl $\rightarrow$ SSM) (S6)
(C_{PS}?msg_mgt_req $\rightarrow$ SLOGIC(msg_mgt_req)
  $\rightarrow$ C_{SP}!msg_mgt_repl $\rightarrow$ SSM) (S7)

Fig.7. Formal description of service customization.

(S6) gives a formal description of service customization by network service interfaces. Programmable Networks are typical technologies of service customization by open application interfaces. So (S6) can be regarded as an abstract description of service customization in Programmable Networks. (S7) gives a formal description of service customization by message passing in the network environment. Active Networks are typical technologies of service customization by delivery of active packets in networks. So (S7) can be regarded as an abstract description of service customization in Active Networks. (S5) in Fig.6 gives a formal description of service customization by management channels. As we know, configuring service out-of-band by management interface is a typical service customization in traditional network systems. So (S5) can be regarded as an abstract description of service customization in traditional network systems.

According to the analysis of (S5), (S6) and (S7), we can conclude that SECUM provides a single formal model describing formally the service customizations of traditional network systems, Programmable Networks and Active Networks respectively. It can provide a unified framework for studying the modern technologies of service customization.

SECUM provides a framework for customizable services, and distinguish theoretically the modes of service customization through management domain, application domain and network domain. In the next section, we will study how to implement service customization within SECUM framework.

3 Component-Based Protocol Implementation Model

In order to show the availability of SECUM in practical network system, we should study the model of implementing SECUM. In this section, a component-based protocol implementation model (CPIM) is proposed for implementing the structure of SECUM.

3.1 Definition of CPIM

The principle of CPIM is to take one type of protocol messages with complete service meaning as a basic component of the protocol. In this way, we can decompose the protocol state machine into sub-machines, in which each sub-machine is associated with sending and receiving each type of protocol message. We call a protocol message with complete service meaning a transactional message. It can be combined with state sub-machine to form a message object class, which is represented by MSG. The state machine in MSG corresponds to PSM in SECUM.

Data structure that is used to storing states of services can be combined with logic of service provided by the protocol to form a service object class, SRV, which is service customization related class. Different MSGs interact with each other through SRV to fulfill a service. The state machine in SRV corresponds to SSM in SECUM. Taking the same assumptions as those for SECUM, CPIM is also an implementation model for single protocol layer. So CPIM defines a communication interface object class, COM, which encapsulates the data structure and processing functions for underlying message transferring. CPIM defines an application interface object class, APP, which encapsulates the data structure and processing functions for transferring service primitives with upper users. The state machine in APP corresponds to ISM in SECUM.

The structure of CPIM is illustrated in Fig.8. MSG encapsulates message related state machine, which is used for protocol state transition when sending and receiving messages of the protocol. SRV encapsulates states and logic of services provided by the protocol. COM encapsulates data and logic for invoking underlying protocol to transfer messages of the protocol. APP encapsulates states and processing logic for exchanging service primitives with application processes.

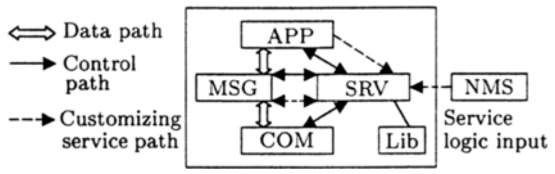

Fig.8. CPIM structure.

There is one data path both between APP and MSG and between MSG and COM respectively, which is used for transferring data transparently through the protocol layer. There are three control paths between SRV and APP, MSG, and COM, which are used for coordinating operations to execute service logic between the SRV and other three classes of CPIM. In addition, there are three paths of service customization between SRV and APP, MSG and outside network management station, NMS, which are used for inputting and modifying service logic of SRV to fulfill service customization.

In order to validate CPIM, it is necessary to show that protocols with transactional messages do exist. In fact, many application protocols in

Internet are transaction oriented protocols, such as HTTP (see RFC 2616) and SIP (see RFC 2543), in which each type of messages specified is a command of meaningful service request.

3.2 Discussions of CPIM

CPIM can be used not only for implementing a new protocol with the abilities of service customization, but also for adding the abilities of service customization in an existing network system. Fig.9 shows an application case for an existing network system, in which APP and SRV are moved out of protocol entity to expand original implementation with some service customizing abilities.

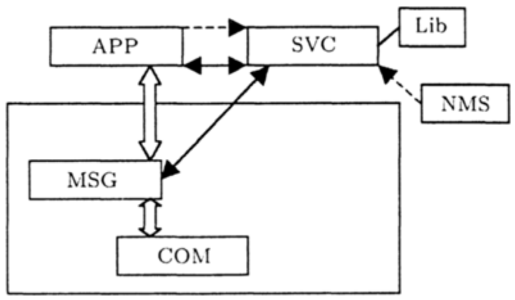

Fig.9. CPIM application structure.

Another application of CPIM is to generate some complex network system by composing several service components within SRV. This kind of application was proposed by Larry Peterson *et al.* in [17]. The difference between these two approaches is that CPIM is based on current network architecture and compatible with current network systems.

4 Examples of Applications in SECUM and CPIM

In the following subsections, some examples of applications of SECUM and CPIM are shown to illustrate the availability of SECUM and CPIM.

4.1 A Customizable IP Service Model

The differentiated service (DS) model is the most practical model of IP QoS-enabled service. The important issue of service customization in differential service model is to create and deploy the Per-Hop Behavior (PHB). In SECUM, PHB can be regarded as a message component used for protocol message interaction. Different PHBs store different logics of transferring packets in IP router, which determines the IP service. According to SECUM (see intermediate node model without service interface illustrated in Fig.2), packet processing in an IP router consists of message interaction (that is, IP message composition and DS extension composition) and message forwarding. PHB defines the message forwarding logic that determines the characteristic of IP service.

According to the IETF's definition of differentiated service model, an accepted PHB should be defined by a standard protocol. Creating a PHB should pass through a standardizing process, such as designing prototype, proposing standard draft, validating the draft by a half or one year, being approved by the standard authority and being deployed by service providers. This process is too long to meet the requirements of different IP services in future, especially the requirements of user customizable IP service.

By applying SECUM, a service customizable differentiated service model can be designed, with which, users can directly customize their own PHB without going through the standardization process.

First, a set of standardized PHB should be defined. The packet scheduling algorithms corresponding to this set of PHB should be able to be composed into a meaningful algorithm of packet scheduling. Looking at the existing algorithms of packet scheduling can show that the composition of packet scheduling algorithms is available. For example, RIO[18] algorithm is in fact composed of two RED[19] algorithms to provide Assured Forward PHB in differentiated service. The problem of composition of packet scheduling algorithms is beyond the scope of this paper.

Then, the packet encapsulation protocol for differentiated service should be expanded to support service customization. The extension includes defining unique identifying mechanism of user-customized services and adding a header extension field to store user-customized service logic. In the standard of PHB (see RFC2474), DSCP (DS Code Point) is represented by 6 bits, which cannot identify uniquely a differentiated service customized by an end user. The unique identifying mechanism for customizable differentiated service is defined as follows. The expanded encapsulation protocol will identify the differentiated service by checking both DSCP and the source identification carried in a packet, not by checking only the DSCP as defined in standard DS model. Each PHB supported by a node should have one item in the Service Logic Table (SLT) stored in the node. An item of SLT includes three fields: DSCP, Source Identifier (SID), and service logic (SL). The SID can simply be an IP source address. If the DSCP represents a standardized PHB, the SID field can be empty. A header extension field can be defined to store the logic of the user-customized service.

Finally, the PHB implementation model in each differentiated service node should be expanded. This extension is to add service logic components corresponding to each standardized PHB into the node of differentiated service and to create an executing environment to run the service logic in the node. This extension can be regarded as implementing PHB in the CPIM model. In this PHB implementation model, each DSCP will correspond to one differentiated service logic, not directly to a PHB any more. For the standardized differentiated service, the service logic is the invocation of service logic component corresponding to the standardized PHB. For the user-customized differentiated service, the service logic is a procedure of calling a set of service logic components corresponding to a set of standardized PHBs. This procedure can be interpreted and executed in the executing environment.

The scenario of service provision in this service-customizable IP service model (see Fig.10) is as follows.

1) A user or a network manager customizes his own differentiated service in the source node by using User-end Service Customization module (USC). USC will generate a DSCP that is unique within the node for identifying the customized service if the customization process is successfully completed.

2) The Service Logic Generation module (SLG) will transform the user-customized service into a valid service logic.

3) When the user sends a message in his customized service, source node encapsulates the DSCP and service logic corresponding to the service into the packets carrying the message.

4) When a node supporting customizable differentiated service receives a packet, it will look up the Service Logic Table (SLT) with the values of DSCP and SID in the packet (note: SID can be set to the source IP address). If there is an item in SLT corresponding to the packet, then the service logic stored in the item is invoked. If there is no item corresponding to the packet, but there is service logic extension part in the packet, then a new item in SLT should be established to store the new PHB information. If there is no item in SLT corresponding to the packet and no service logic extension part in the packet, then the packet will be processed by default PHB routines.

5) The packet of user-customized differentiated service will be forwarded in the rules customized by the user in all nodes supporting the customizable IP service along the route to its destination. In this way, the end-to-end IP service customization is realized.

This approach differs from the approaches of Active Networks or Mobile Agent in the following. Firstly, there is no active packet that should carry code in its data field. Secondly, there is no execution environment in each node established specially for executing active packet or mobile code. This approach makes full use of service logic separation structure in SECUM and component-based protocol implementation model of CPIM, and transforms the differentiated service model into a customizable differentiated model. SECUM is an internal service customization model, while the Active Networks are external service customization models that can only be used upon the existing network services.

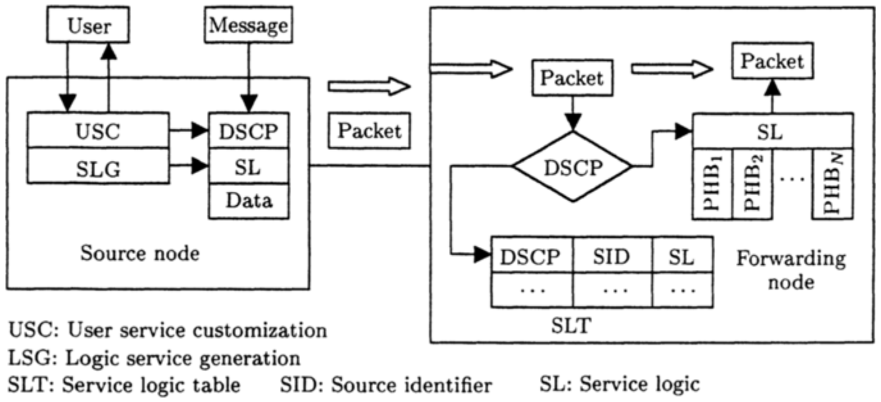

Fig.10. Customizable IP service model.

4.2 SIP Implementation

SIP is a protocol used for creating, modifying and terminating a multimedia call (also called a session), which has been widely used or accepted in multimedia conference in Internet, IP telephony and 3G mobile telecommunication networks. SIP is transaction-oriented protocol that provides six basic request methods (or message types): INVITE, ACK, BYE, CANCEL, REGISTER, and OPTION. SIP is designed with the ideas of service customization. It focuses on designing basic service components, not just service itself. So applications can combine SIP service components to form the specific service required by application. It is easy to extend the functions of SIP. For example, PINT (see RFC 2848) has extended SIP for providing service of calling PSTN subscriber in Internet with three extended request methods: SUBSCRIBE, NOTIFY and UNSUBSCRIBE.

We have implemented User Agent part of SIP according to the CPIM. In our SIP implementation, INVITE, ACK, BYE, CANCEL, REGISTER, and OPTION are defined as a message object class respectively. The response message corresponding to each of these request methods is also defined as an object class, sharing one protocol state machine with its corresponding method. We define a call state object class, CALL_STA, to associate the operations issued by these message classes to provide specific services required by applications. CALL_STA here corresponds to SRV in CPIM. We can put the service policies in CALL_STA to customize specific service. Testing on our SIP implementation shows that CPIM is available in implementing protocol with some abilities of service customization.

5 Related Work and Analysis

Programmable Networks have proposed several excellent frameworks for service customization[5,6,20]. A. Campell *et al.* have associated Active Networks with Programmable Networks in [6]. But there is no formal model for service customization to clarify so many concepts. The formal SECUM proposed in this paper gives a simple and clear explanation to the mechanism of service customization used in Programmable Networks and Active Networks.

JAIN[7,8] is trying to establish a general signaling protocol interface over Public Switched Telephone Networks, Internet and Mobile Networks. It pays much attention to specifying the Application Programming Interface (API) in the Java environment, but does not concern the implementation mechanism of service customization. In fact, the goal of JAIN is to define only standardized API used in industry. It pays little attention to the mechanism of service customization that can be implemented in current network systems. Our work pays more attention to the mechanism used for implementing service customization.

Larry Peterson *et al.* proposed a dynamic protocol model[17] in 1992. It is a very flexible network architecture, which can combine the required service with micro-protocol layers. But this model is very different from the current network architecture and there will be lots of work to do for it to be applied to practical network systems. Both SECUM and CPIM proposed in this paper are based on the current network architecture and they are compatible with practical network systems used today.

X. Logean *et al.* proposed some formal techniques used for describing network service[21]. They argue that it seems more beneficial (cost-effective and faster) for formal techniques to focus on the actual implementation, or at abstraction levels that accurately represent the real implementation. We agree with their opinions. But we think that applying formal techniques to concept model is also very important in research on new technologies, such as Active Networks.

C. Partridge *et al.* proposed a service customizable routing approach, called FIRE: Flexible Intra-AS Routing Environment[22]. A traditional routing protocol generates a single forwarding table at each router, which the router then uses it to determine where to forward incoming traffic. FIRE extends that notion by generating a set of forwarding tables, each uniquely defined by three pieces of information: the algorithm used to compute the table, the properties used by the algorithm in its computations, and a packet filter that determines which class of traffic uses the forwarding table. In FIRE, all these variables can be configured by the network operator at run time. This approach adopts some ideas similar to SECUM, such as component-based forwarding tables for different classes of traffic. But FIRE is not a general model of service customization. We believe that SECUM can be used to optimize the FIRE approach.

Compared with the related studies, SECUM and CPIM have the following advantages.

SECUM is a conceptual model for service customization based on hierarchical architectures of existing networks. So SECUM can be applied directly in the existing networks, including telecommunication networks and Internet.

SECUM provides a unified model of service customization for Active Networks, Programmable Networks and traditional networks. As we know, Active Networks customize the service by active packet, Programmable Networks customize the service by application programming interfaces and traditional networks customize the service by management channels. But we do not know their relationship in service customization. According to SECUM, these service customization techniques only customize externally the services of network systems in different domains. Active Networks customize service in network domain, Programmable Networks in application domain and traditional networks in management domain.

SECUM provides a conceptual model of internal service customization. This is the main difference between SECUM and existing techniques of service customization. We believe that the model of internal service customization can provide more powerful abilities of service customization for Next Generation Networks.

CPIM is proposed to show that SECUM can be implemented in existing networking environments. However, there is no implementation model in the existing networks for the dynamic network architecture proposed by L. Peterson.

The disadvantage of SECUM is that it needs to implement again the protocols for customizing the services provided by the protocols. It may be a costly work to make network service customizable by SECUM if there is no new protocol introduced into the networks. In this case, the Active Networks and Programmable Networks may be good choices. If there are some new protocols to be introduced into networks to realize some new services, the SECUM can be applied to the networks to enable the service customization. In this case, the SECUM may be a better choice.

We have noticed that the component-based software architecture is very critical for opening the services of some software module. As Ali Arsanjani says[23]: "Some software module needs to provide the services. Attacking this problem in an ad hoc fashion by merely exposing services without adequately defining underlying component boundaries and responsibilities will lead to performance and scalability problems as well as extremely volatile interfaces as the projects unfold. Thus, a component-based software architecture provides a very stable and yet flexible foundation for a service-oriented architecture." SECUM is a component-based software model in the view of implementation. The above statements can support our work on SECUM and CPIM.

6 Conclusion

In this paper, we propose an abstract concept model, SECUM, for service customization based on current network architecture in view of interaction between protocol and service. We give a formal specification of SECUM so that it can be used to describe and analyze the abilities of service customization of current active technologies, such as Active Networks and Programmable Networks. We propose a customizable IP service model to show its applicability in the existing networks.

We also propose a component-based protocol implementation model, CPIM, according to the concepts of SECUM. CPIM can be used as an implementation model for realizing the mechanism of the service customization proposed in SECUM. We describe our work on the implementation of SIP to show the availability of CPIM.

Compared with the existing service-customization techniques, SECUM is a service customization model internal to network system that may provide more powerful abilities of service customization.

Acknowledgement Zai-Long Zhang and Jia-Gao Wu, who are the members of our research group in the Nanjing University of Posts and Telecommunications, took part in the project of implementation of SIP. We are very grateful for their work on SIP implementation.

References

[1] David L Tennenhouse, Jonathan M Smith, W David Sincoskie *et al.* A survey of active network research. *IEEE Communication Magazine,* January 1997, 35(1): 80–86.

[2] Kenneth L Calvert, Samrat Bhattacharjee, Ellen Zegura, James Sterbenz. Directions in active networks. *IEEE Communications Magazine,* Oct. 1998, 36(10): 72–78.

[3] Jonathan M Smith, Kenneth L Calvert, Sandra L Murphy *et al.* Activating networks: A progress report. *Computer,* April 1999, 32(4): 32–41.

[4] Danny Raz, Yuval Shavitt. Active Networks for efficient distributed network management. *IEEE Communications Magazine,* March 2000, 38(3): 138–143.

[5] Aurel A Lazar. Programming telecommunication networks. *IEEE Network,* September/October 1997, 11(5): 8–18.

[6] Campbell A T, De Meer H G, Kounavis M E *et al.* A survey of programmable networks. *Computer Communication Review,* April 1999, 29(2): 7–23.

[7] John de Keijzer, Douglas Tait, Rob Goedman. JAIN: A new approach to services in communication networks. *IEEE Communications Magazine,* January 2000, 38(1): 94–99.

[8] Ravi Raj Bhat, Rajeev Gupta. JAIN protocol APIs. *IEEE Communications Magazine,* January 2000, 38(1): 100–107.

[9] Simon Beddus, Gary Bruce, Steve Davis. Opening up networks with JAIN parlay. *IEEE Communications Magazine,* April 2000, 38(4): 136–143.

[10] Vu Anh Pham, Ahmed Karmouch. Mobile software agents: An overview. *IEEE Communications Magazine,* July 1998, 36(7): 26–37.

[11] Gregor V Buchmann, Carl A Sunshine. Formal methods in communication protocol design. *IEEE Trans. Communications,* April 1980, COM-28(4): 624–631.

[12] David D Clark. The design philosophy of the DARPA Internet protocols. *Computer Communications Review,* Sept. 1988, 18(4): 106–114.

[13] Jeffrey M Bradshaw, Niranjan Suri, Alberto J Canas *et al.* Terraforming cyberspace. *Computer,* July 2001, 34(7): 48–56.

[14] Gregor V Buchmann. Specifications of a simplified transport protocol using different formal description techniques. *Computer Networks and ISDN Systems,* 1989/1990, 18: 335–377.

[15] Hoare C A R. Communicating Sequential Processes. Prentice Hall International, Englewood Cliffs, New Jersey, USA, 1985.

[16] Roscoe A W. The Theory and Practice of Concurrency. London: Prentice Hall, 1998.

[17] O'Malley S W, Peterson L L. A dynamic network architecture. *ACM Trans. Computer Systems,* May 1992, 10(2): 110–143.

[18] David D Clark, Wenjia Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Trans. Networking,* August 1998, 6(4): 362–373.

[19] Sally Floyd, Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Networking,* August 1993, 1(4): 397–413.

[20] Campbell A T, Kounavis M E, Villela D A *et al.* Spawning networks. *IEEE Network,* July/August 1999, 13(4): 16–29.

[21] Xavier Logean, Falk Dietrich, Jean-Pierre Hubaux *et al.* On applying formal techniques to the development of hybrid services: Challenges and directions. *IEEE Communications Magazine,* July 1999, 37(7): 132–138.

[22] Craig Partridge, Alex C Snoeren, W Timothy Strayer *et al.* FIRE: Flexible Intra-AS routing environment. *IEEE Journal on Selected Areas in Communications,* March 2001, 19(3): 410–425.

[23] Ali Arsanjani. Developing and integrating enterprise components and services: Introduction. *Communications of the ACM,* October 2002, 45(10): 30–34.

Su-Bin Shen received the B.Sc., the M.Sc. and the Ph.D. degrees in computer science and engineering from Southeast University, China, in 1984, 1989 and 2000 respectively. He is currently with the Research Center of Network Technology, Nanjing University of Posts and Telecommunications, China, as a research professor. He is the author of more than 20 papers in referred journals and conferences. His research interests include network architecture, protocol modeling, next generation networks, network security and telecommunication software.

Guan-Qun Gu received the B.Sc degree in computer science and engineering from Nanjing Institute of Technology (which is now renamed as Southeast University), China, in 1962. He is currently with the Department of Computer Science and Engineering, Southeast University, China, as a professor. He is an academician of the Chinese Academy of Engineering. He currently holds the president of Southeast University. He has published more than 100 papers in referred journals and 6 books on computer networking and applications. His research interests include computer networks, computer-integrated manufacture systems, network service platform and network applications.

Shun-Yi Zhang received the B.Sc degree in electronic engineering from University, Tianjin, China in 1968. He currently holds the vice-president of Nanjing University of Posts and Telecommunications. He is also a professor in the Research Center of Network Technology, Nanjing University of Posts and Telecommunications. He has published more than 30 papers in referred journals and conferences. His research interests include computer communications, network optimization models and algorithm, service management in telecommunication system and management information systems.