16
Formal Description of Services*

Yuzhang Liu, Fangchun Yang, Junliang Chen
Beijing University of Posts & Telecommunications
BOX 206#, Beijing, 100088, P.R. of China
Tel: +86 10 2282007, Fax: +86 10 2022770
E-mail: yzliu@bupt.edu.cn

Abstract

On the basis of reviewing the basic concept of IN and life cycle of the service, this paper gives the concept of SLB, and formally describes the service in the syntax, behavior, and semantics way, which provides an approach to resolve the service interaction in the service specification phase.

Keywords

IN, SCE, Service Interaction.

1 INTRODUCTION

Service is a concept of Intelligent Network that the telecommunication network provides to the users. A service provides stand-alone functionality for some commercial purposes.

The aim of IN is to introduce services quickly and deploy the services onto the network rapidly. To realize this, ITU-T defines four layered IN Concept Model (INCM). The first plane of INCM is the service plane. The service plane represents an exclusively service-oriented view. The IN service is also called the supplement service which is based on the POTS (Plain Old Telephone Service) and adds value to customers. The second plane is the global functional plane (GFP), where the service independent building block (SIB) is defined and used to construct a service.

In the IN recommendation CS1 (Capability Set 1), ITU-T has defined 25 services, some of them are widely used, such as Freephone service, Account Card Calling service, Call Forwarding service, and Originating Call Screening service etc.

The service has its own life cycle (the life cycle of the service is shown in Figure 1.)

* Sponsored by Doctoral Research Fund of the State Education Commission of China.

- Requirement: Specify the requirement and behavior of a service by formal language, such as SDL, MSC, TTCN.
- Creation: Create the service according to the requirement of the service by SIB-based language in Service Creation Environment (SCE), including service testing and service validation.
- Deployment: Deploy the service to telecommunication network, then the software of this service is downloaded to the physical nodes which control and manage this service.
- Subscribed: The service is subscribed by the customers, and the customers customized the service.
- Invocation: The service is activated/deactivated by the users.
- Termination: The service is terminated and withdrawn by network operators.

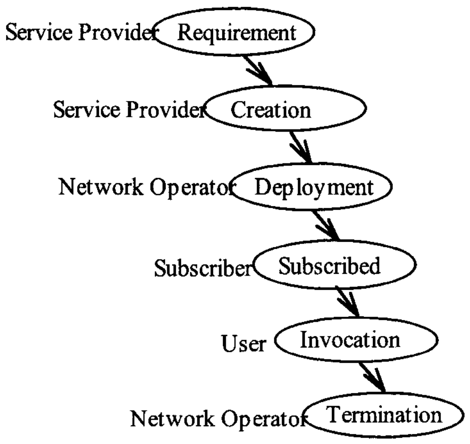

Figure 1 The life cycle of the service

The appearance of the IN technology, with its rapid and abundant service development, has stressed the problem of interactions between services. The service interaction has become a main obstacle of the service creation and IN technology. The difficulty should be resolved in such way:

- Formal definition of the services, including the syntax, behavior and semantics of the services in the requirement and creation phases.
- Dynamic negotiation in the deployment, the subscribed and the invocation phases.

This paper proposes a method to formally describe the service from the syntax, the behavior and the semantics views and provides an approach to handle the service interaction at the specification level. The method evolved from an IN project at BUPT IN research supported by the 863 project of China. We have implemented an SCE tool to create services by SIB-based language and successfully download these services into SCP to execute. Now we are engaged in the research on the service validation, detecting and resolving the service interaction.

2 THE CONCEPT OF SERVICE LOGIC BLOCK AND THE EXECUTION OF SERVICES ON IN

The IN service can be created by the SIB-based language. One service is constructed by several executable chains of SIBs. Among those SIBs, there is a special SIB called BCP (Basic Call Process) which provides the basic call processing, implements the function of connecting and disconnecting the call at proper time and holding the call instance data for call processing. The BCP is described with a finite state machine, the basic call state model (BCSM). The BCSM consists of states that represent the processing done by exchanges, transitions and detection point. Detection Points (DP) are points in call processing where the SSF can determine if a request to the SCF should be reported according to the service control logic. The detection points have two types, trigger detection point (TDP) that is the static point and armed when the service is deployed and event detection point (EDP) that is the dynamic point and armed when the service is executed.

The service logic program (SLP) described by SIB-based language is shown below:

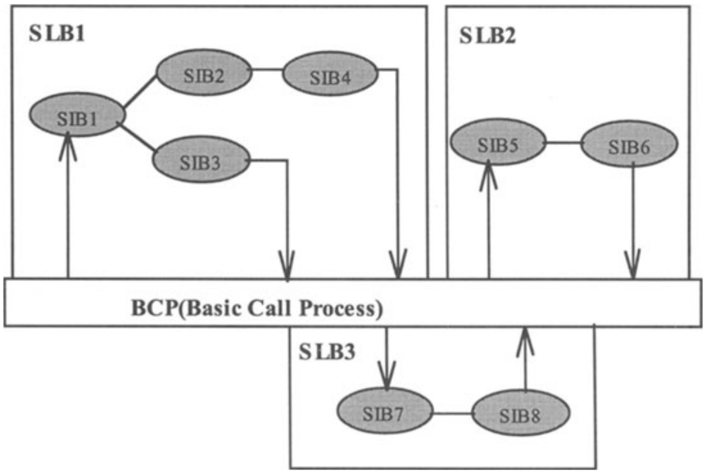

Figure 2: The Service Logic Program of one Service

The SLP of a service consists of several SLBs (Service Logic Block). The SLB is a chain of SIBs that starts from BCP and finally returns to BCP (i.e. in Figure 2, the service has three SLBs). Each SLB of a service is invoked by one detection point, either TDP or EDP. The relationship between O_BCSM (originating BCSM) and SLB is shown in Figure 3.

During the execution of a service, control is switched between SSF/CCF and SCF. Firstly the call is processed in SSF/CCF, when a detection point is encountered, the control is switched to SCF and one SLB of this service is executed, after the SLB is finished, the control returns to SSF/CCF.

The introduction of SLB has following advantages.

- Subdivide the function of a service, such as the Freephone service (shown in Figure 4) has two SLBs. One SLB implements the function of One Number (one feature of the Freephone service), and the other SLB implements the function of Reverse Charge (the other feature of the Freephone service).
- SLB can be used to handle the service interaction. The interactions between services always come from the interactions between the SLBs of different services, such as the Freephone service and Account Card Calling, the two services have the SLBs which are invoked from the same EDP (O_Disconnect).

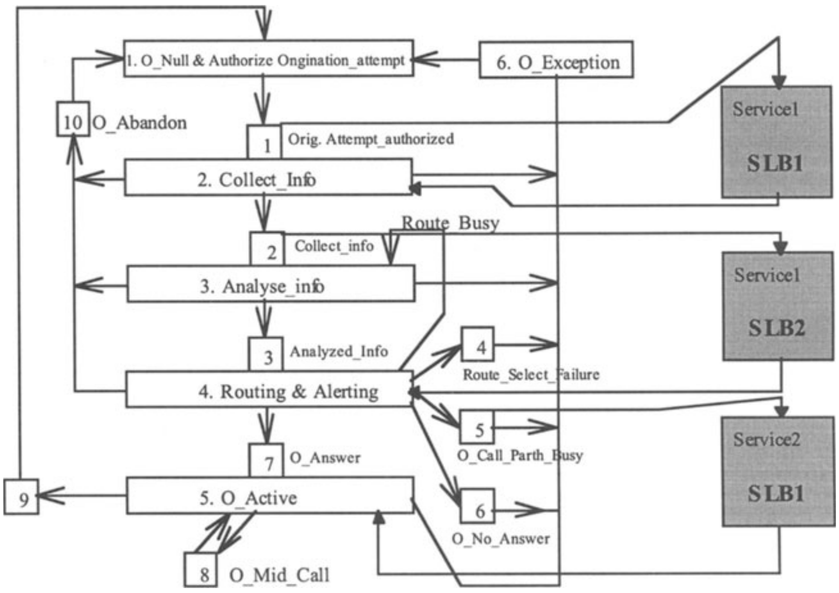

Figure 3 The relationship between O_BCSM, Services and SLBs

The SLB of a service is different from the feature of the service. The feature describes part of the functionality of the service, it can hardly be described by SIBs, while the SLB describes the part of execution of the service (always also represents the part function of the service) and it can be formally represented by SIBs (a chain of SIBs which starts from BCP and returns to BCP), so we can define:

$$\text{Service} = \sum \text{SLBs}$$

In our IN project, we realize ten SIBs defined by ITU-T in CS1 (*Algorithm*, *Charge*, *Compare*, *Log*, *Screen*, *SDM*, *Translate*, *UI*, *Verify*, *BCP*), and realize three SIBs defined by Australia (*Connect*, *EDPRequest*, *ReleaseCall*), and we implement two SIBs defined by ourselves (*Start* SIB used to choose the number from the whole number that user dials; *Charge2* SIB used to charge dynamically).

A typical example of IN services is the Freephone service which is widely used all over the world. The SLP of the Freephone service in our IN product is shown in Figure 4. The *Start* SIB chooses the formal number from the whole number that is reported from BCP; and *EDPRequest* SIB arms the EDPs (O_Abandon, O_Disconnect). *Screen* SIB examines if the formal exists in SDF; if NoMatch, *UI* SIB notifies the user that the number does not exist and *ReleaseCall* SIB disconnects the connection; if Match, *Translate* SIB translates the formal number into a destination number. *Queue* SIB creates and maintains a queue for every destination number invoked, if the destination number is free, then *Log* SIB records the call data, and *Connect* SIB connects the call; if the queue is full or the time expires, *UI* SIB notifies the user and *ReleaseCall* SIB disconnects the connection.

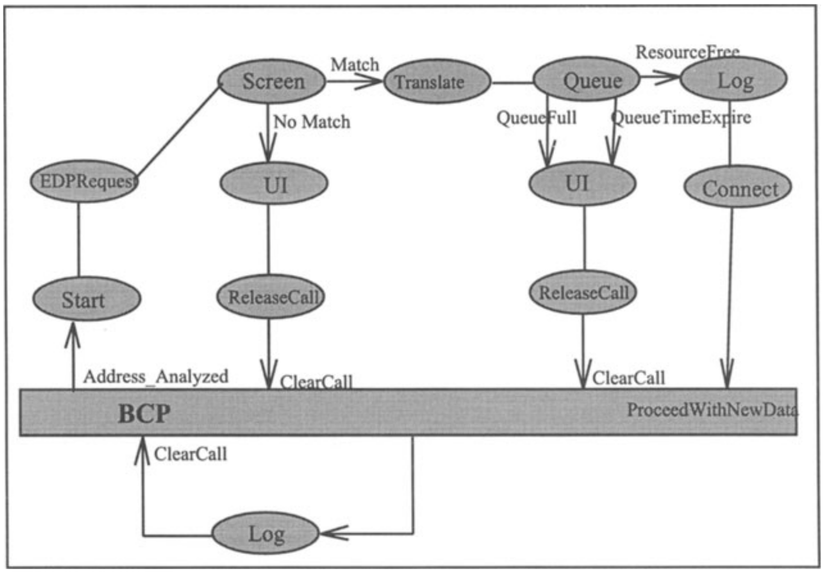

Figure 4 The SLP of the Freephone service

During the execution of the Freephone, firstly the user dials the prefix number 800 and formal number (for instance 12345), the number is analyzed by CCF and it is sent to SCF as a TDP. In SCF, an SLP is invoked. This SLP arms other DPs, retrieves the formal number from

SDF. If the formal number does not exist, an announcement is sent to the user; otherwise the SDF translates the formal number into a destination address according to the location of caller and the time, the SLP is hung and the control returns to CCF. The call continues, while the user is connected to the destination address and starts conversation. At the end of the conversation, when the user hooks on, an event is captured by SSF/CCF and sent to SCF as a detection point. In the SCF, the SLP is resumed to handle the charging (reverse charge). During the execution of the Freephone service, the control is switched between the SSF/CCF and SCF. The control switched between SSF/CCF and SCF is shown in Figure 5.

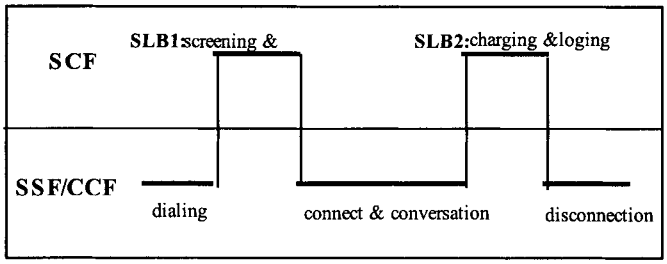

Figure 5 control switched between SSF/CCF and SCF

3 THE SYNTAX OF THE SERVICE

The service is considered as SLP (service logic program, the software processing in SCF) and the database in the SDF. To describe the syntax of a service, it is necessary to describe the syntax of SLP and service-related databases. The SLP is constructed by several chains of SIBs. Each SIB has its own data, including SSD (Service Support Data) and CID (Call Instance Data).

During the execution of a service, the service needs data to record the user information. The data is stored in the service-related database in the SDF. So, the syntax of a service consists of two parts:

Syntax_Service = *SLP* £« *DB*

SLP refers to the service logic program which describes the execution of a service, and *SLP* consists of several executable chains of SIBs (SLB). *SLP* is defined as 5-tuple £° <*Q*, *E*, *O*, *T*, q_0>.

Q refers to the set of SIBs defined in the GFP.

Q = {*BCP*, *Queue*, *Charge*, *Algorithm*, *Verify*, *SDM*, *ReleaseCall*, *Translate*, *UI*, ...}

E refers to the environment of each SIB, which describes the service data of this SIB. The element of *E* is different for different SIBs and also different for the same SIB under different situations.

O is the outlets of the specified SIB, such as the *Screen* SIB, having two outlets: *Match* and *NoMatch*.

T refers to a function: *Q*¡Á*E*¡Á*O* $\rightarrow$ *Q* ¡Á*E*

$q_0 \in Q$, refers to the initiating and terminating states of all service logic program.

DB refers to the service-related database stored in the SDF. In the syntax definition of a service, only the logic profile of the database is defined. The BNF definition of *DB* is as follows:

DB ::= { database_definition $\}^*$
database_definition ::= {database_fields$\}^+$
database_fields = fields_name + fields_type
fields_name = string
fields_type = Integer | String | ...

The definition of SLP and DB of the Freephone service is as below:
The SLP definition of the Freephone service is: $\mathrm{SLP}_{\mathrm{fph}} = < Q_{\mathrm{fph}}, E_{\mathrm{fph}}, O_{\mathrm{fph}}, T_{\mathrm{fph}}, q_0 >$.

Q_{fph} = { *BCP, Start, EDPRequest, Screen, UI, ReleaseCall, Translate, Queue, UI, ReleaseCall, Log, Connect, Log*}
E_{fph} = { *<Start, SSD, CID>, <Screen, SSD, CID>, ...*}
O_{fph} = { *Screen_Match, Screen_Nomatch, Queue_Queuefull, QueueResourcefree, ...*}
T_{fph} is omitted.
$q_0 = BCP$.

The description of databases of the Freephone service is omitted.

4 THE BEHAVIOR OF THE SERVICE

From the user viewpoint, the service is an entity which implements the function of how to subscribe this service, how to connect the calling party to the called party, how to interact with users. In the requirement phase of life-cycle of the service, the behavior of this service can be specified by the service provider. The service provider can describe a service by formal language, such as SDL or MSC. The Freephone service is described by MSC below.

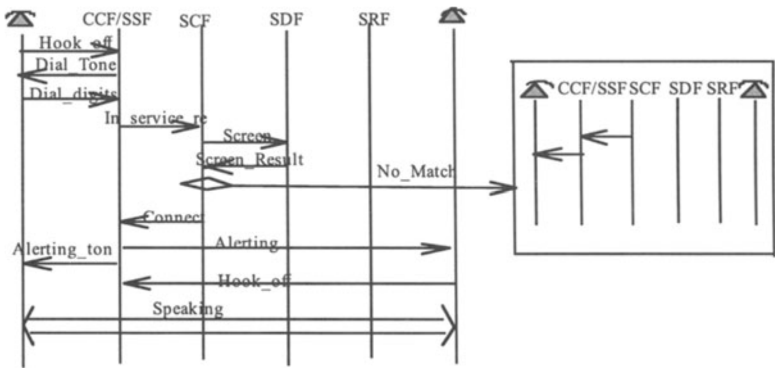

Figure 6 The behavior of the Freephone service described by MSC

For the IN viewpoint, the behavior of a service can be considered as a sequence of information flows between different function entities (FE) and the function entity actions (FEA) which are executed in FEs.

The behavior of a service can be described below £°
Behavior-Service =<*TDP*, *EDP*, *Behavior-Specification*>
TDP refers the trigger detection point of a service.
EDP refers the set of event detection point of a service.
Behavior-Specification is a description of a service based on the SDL language.

The mapping of SIBs in GFP (Globe Function Plane) to DFP (Distribution Function Plane) is the information flows and function entity actions. Figure 7 shows the mapping of SIB in GFP to FEA/IF in DFP.

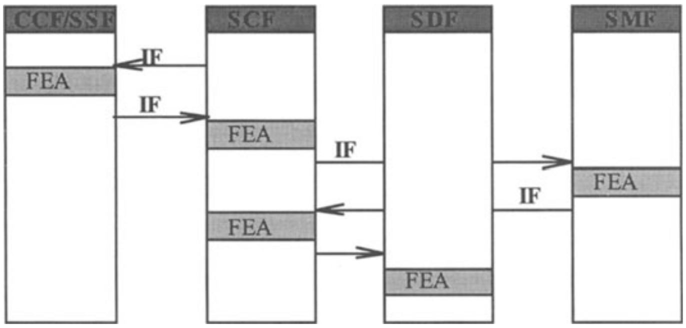

Figure 7 Mapping of SIB in GFP to FEA/IF inDFP

So, we can say: SIB = IF + FEA. IF refers to the information flows between FEs (including SCF, SDF, SSF, SRF, SMF).

We can deduce the *Behavior-Specification* of a service from the syntax of the service automatically by combining the behavior of each SIB, and the *TDP* and *EDP* can also be deduced from the outlets of *BCP* SIB in the *SLP*, so we can deduce the service behavior from the syntax of the service.

The behavior of a service shows the function of the service, and it is useful for the service simulation, validation and detection of service interaction.

- In the service simulation, we can simulate each FE by a process and simulate the FEAs of each SIB by software, according to the IF between different FEs, and the execution of a service can be simulated. The service provider can tell if the execution of a service is identical with his expectation.

- In the service validation, we can compare the behavior of a service written by the service provider in the specification phase to that deduced from the syntax, and tell if they are identical.
- In detecting the service interaction, the detection points and SDL specification are also useful to detect the interaction between services. For example, the Call Waiting service and Call Forwarding on Busy service have interactions because those two services have the same trigger detection point.

5 THE SEMANTICS OF THE SERVICE

The semantics of a service describes the purpose of the service, or the purpose of subscriber to subscribe this service. For example, the subscriber of the OCS (Originating Call Screen) service wants to screen automatically all numbers on his screen list from his handset. The subscriber of the Call Forwarding service wants to redirect his incoming call to the specified number.

The semantics of a service may separate or partly separate from the syntax or behavior of the service. It is impossible to be deduced from the syntax of a service automatically. It must be written by the service provider when the service is created. The semantics of a service can be described by the assertion of logic.

For example the semantics of the service Call Forwarding and OCS (Originating Call Screen) are described below. Call Forwarding is the service that the subscriber can redirect the incoming call to another number when he is busy. OCS service is the service that the subscriber can screen the outgoing call by the screen list.

The semantics of Call Forwarding is:

$$\{\, \exists x\, \forall y\, \forall z\, (\mathrm{Call}(y, x) \wedge \mathrm{Busy}(x) \wedge \mathrm{Reroute}(x, z) \rightarrow \mathrm{Call}(y, z)) \,\}.$$

The semantics of OCS is:

$$\{\, \exists x\, \forall y\, (\mathrm{Screen}(x, y) \wedge \mathrm{Call}(x, y) \rightarrow \mathrm{Block}(y)) \,\}.$$

There should be some basic assertion which describes the POTS service, such as the proposition of Call(y, x) means the user y makes a call to x; Busy(x) means user x is busy; Reroute(x, z) means that call to x is redirected to z. The new assertion used to describe the characteristic of new services must be defined by some basic concepts, such as Screen(x, y) means number y is screened on x's handset; Block(y) means the call to y is blocked.

Also, they are some pre_conditions and post_conditions of each SLB. The pre_conditions describe the assumption of the SLB, while the post_conditions describe the execution result of the SLB, such as the modification of call instance data. So the semantics of a service can be described below:

Semantics = *ASSERTION* + *CONDITION*
CONDITION = {SLB_Condition}*

SLB_Condition = Pre_condition + Post_condition

The semantics can be used to detect the service interaction. Now we give an example of how to detect an interaction from the OCS service and CF (Call Forwarding) on busy service. The OCS screens all outgoing calls in the screen list of the handset, while CF on busy redirects the incoming call to the specified number when the handset is busy. If user x (his telephone is also x) uses OCS to screen number y in his handset, but another user z can use CF on busy to redirect incoming call to y, and then he dials himself (dials number x) to get the effect to call to y.

To describe the two services, we define a proposition *call*(a, b, c, d), where a refers to the calling number, b refers to the dialed number of caller, c refers to the called number and d refers to the destination number.

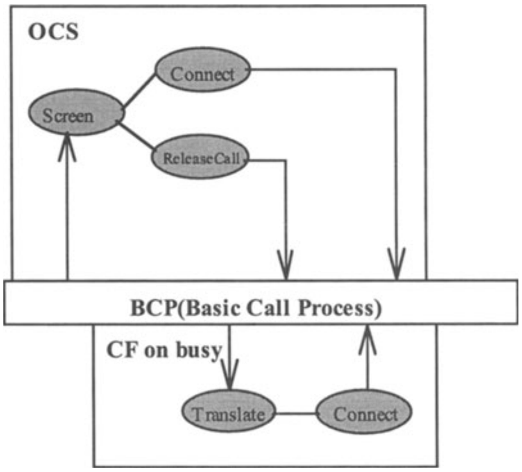

Figure 8 The SLP of the OCS Service and the CF on busy Service

The SLP of the two services is shown in Figure 8. The Pre_condition of OCS is: for the proposition of *call*(a, b, c, d), c = d, the called number must be equal to the destination number. During the execution of the call, user z dials the number x, the proposition is *call*(x, x, x, x), after invocation of CF on busy, the proposition is changed to *call*(x, x, x, y), and when the OCS is invoked, the *call*(x, x, x, y) violates the assumption of OCS. Thus, the interaction is detected.

6 CONCLUSION

IN gives approach to rapidly and economically create services. As everyone, every company can create services. It is difficulty to deploy the services from different venders into the whole telecommunication network and make them co-operatable, and it also adds the difficulties for the service management and resolving the service interaction. The formal description of services, in the syntax, behavior, semantics way can standardize the service in the specification and creation phases. This may help for the services cooperating from different venders and different network components, and provides a way to resolving the service interaction, also it can support user for the service simulation and service validation in the service creation phase in SCE.

7 REFERENCES

A ITU-T, CS-1, Recommendation Q 12XXseries.

B Feature Interaction In Telecommunication System, L.G. Bouman and H.Velthuijsen, IOS Press.

C A Practical Approach to Service Interaction, Eric Kuisch, IEEE Communication Magazine, August 1993.

D Intelligent Network Service Creation Environment, Yuzhang Liu, ISIB'95, Beijing, China.

E The Research and Implementation of Intelligent Network, Wang Bai, Doctoral Thesis of BUPT, P.R. of China, 1995.

8 BIOGRAPHY

Yuzhang Liu received a Bachelor degree in computer science in 1992 from the university of Tianjin, China, and graduated from Beijing University of Posts and Telecommunications (BUPT) and received the Ph.D. degree in electronics and communications science in 1996. From 1994 to 1996, he takes part in project of Intelligent Network in BUPT, and have developed the tool of Service Editor, Service Analyzer and IN Emulator which can emulate the whole IN system and service execution. His research interests include Intelligent Network, Feature Interaction, Software Engineering.

Fangchun Yang received the MS in computer science, Ph.D. in communication and electronic system from BUPT in 1986 and in 1990 respectively. Currently, he is a professor of National Lab of Switching Technology and Telecommunication Networks at Beijing University of Posts and Telecommunications (BUPT). His Research interests include intelligent network, software engineering. He has been responsible for more than ten national projects and collaboration abroad since 1990. At the moment, he is in charge of a national key project to develop IN project in China.

Junliang Chen graduated from the Department of Telecommunications, JiaoTong University, Shanghai, China, in 1955. From 1957 to 1961, he studied as a graduate student at the Moscow Institute of Electrical Telecommunications, Russia. From 1979 to 1980, he was a visiting scholar at the Departments of Electrical Engineering and Computer Sciences of the University of California, Berkeley. From 1980 to 1981, he was a visiting scholar at the Department of Computer Sciences of the University of California, Los Angeles. In 1989, he also visited the University of Bristol. Currently, he is a professor at the Beijing University of Posts and Telecommunications (BUPT), and a member of the Chinese Academy of Sciences and Chinese Academy of Engineering. His research interests include switching systems, telecommunication networks, software, and fault-tolerant computing.