Dai Y, Yang L, Zhang B. QoS-driven self-healing web service composition based on performance prediction. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 24(2): 250–261 Mar. 2009

QoS-Driven Self-Healing Web Service Composition Based on Performance Prediction

Yu Dai[1] (代 钰), *Member*, *CCF*, Lei Yang[2] (杨 雷), and Bin Zhang[2,*] (张 斌)

[1] *College of Software, Northeastern University, Shenyang 110004, China*

[2] *College of Information Science and Engineering, Northeastern University, Shenyang 110004, China*

E-mail: neu_daiyu@126.com; {yanglei, zhangbin}@ise.neu.edu.cn

Received March 17, 2008; revised November 27, 2008.

Abstract Web services run in a highly dynamic environment, as a result, the QoS of which will change relatively frequently. In order to make the composite service adapt to such dynamic property of Web services, we propose a self-healing approach for web service composition. Such an approach is an integration of backing up in selection and reselecting in execution. In order to make the composite service heal itself as quickly as possible and minimize the number of reselections, a way of performance prediction is proposed in this paper. On this basis, the self-healing approach is presented including framework, the triggering algorithm of the reselection and the reliability model of the service. Experiments show that the proposed solutions have better performance in supporting the self-healing Web service composition.

Keywords service composition, QoS, self-healing, performance prediction, reselection

1 Introduction

With the popularity of Web services, service composition[1] which integrates different Web services from different service providers has become one of the major challenges in the area of Service Oriented Computing (SOC)[2]. Web services operate autonomously in a highly variable environment (the Web). Accordingly, the QoS of Web services may change with a relatively higher frequency[3] and such dynamic nature of the services will influence the runtime performance of the composite service.

Currently, many people focus their study on how to select services according to their QoS[4,5] in order to make the composite service meet users' constraints and exhibit better runtime quality. In their studies, QoS of the component services rely on the estimation of the service execution parameters. However, in the execution, the actual QoS is sure to deviate from the estimated one, for example, because of the network load. Such dynamic nature of Web services requires that the composite service must have self-healing ability, which means that composite service can heal itself if any execution problem occurs, in order to complete its execution successfully and meet the users' constraints.

The easy solution to the problem is to re-select the services every time a change occurs. However, it is not feasible due to the high time complexity of the reselection[6], which will interrupt the execution of the composite service and influence its performance. For the purpose of minimizing the extra delay caused by reselection, approaches[7,8] offering the solution are proposed, that is to back up a composite service for each component service. Then, when a component service incurs a failure, the composite service can easily switch to a replacement and the self-healing will not affect the execution performance of the composite service. Since these approaches are to find the replacement composite service in the selection, the replacement may not be available as a result of QoS change of component service in the replacement. This will lead to another reselection that may in turn lead to an extra delay and affect the performance of the composite service.

With all these problems in mind, we present a solution for the composite service to heal itself. Such solution is an integration of approaches to backing up in selection and reselecting in execution. The key to our approach is performance prediction that makes the composite service heal itself from failure as quickly as possible and minimizes the number of reselections. Firstly, through performance prediction, the component services which will have a QoS violation

Regular Paper

*Corresponding Author

This work is supported by the National Natural Science Foundation of China under Grant No. 60773218.

can be predicted, and if the corresponding replacement composite service is not available, the reselection aiming at finding the new replacement will be triggered as early as possible. In this way, it will be more likely to make the reselection complete before the invocation to the component service and thus, the composite service can heal itself from the failure as quickly as possible. Secondly, by the performance prediction, the reliability of services can be modeled and quantified, on the basis of which QoS-driven reselection can find a replacement composite service with higher reliability and better QoS. This will make the replacement composite service more reliable and minimize the number of reselections. All in all, by performance prediction, the composite service can be more effective in meeting the demands of the dynamics of Web service.

The contribution of this paper includes: 1) a self-healing approach based on performance prediction is proposed; 2) a method of performance prediction based on semi-Markov model is presented; 3) on the basis of performance prediction, a service reliability model which integrates processing and transmission reliability is proposed and its corresponding reselection algorithm is also presented. Experiments show that the proposed solutions have better performance in supporting self-healing Web service composition.

2 Related Work

In this section, we will review the work done in self-healing Web service composition and service reliability.

The service reliability is the probability of the degree to which a request is correctly served by the service within the estimated QoS. In the current studies the service reliability is seldom addressed. In [9], the researchers use the successful execution rate of a service to quantify service reliability. In [10], the author uses Poisson Distribution to quantify the reliability of grid service with the assumption that the data transmission speed and failure rate of the network are constant. However, such an assumption does not always hold true. Comparing with the above work, we model the service reliability as an integration of request processing reliability and transmitting reliability. To quantify processing reliability, a common component service's reliability model[11] is used. The data transmission speed relies on the state of the network at the predicted time, the duration of the state, and the interval from the predicted time, e.g., today, to the time to be predicted, e.g., tomorrow. According to such nature of data transmission speed, we quantify the transmitting reliability by performance prediction, which is based on semi-Markov model.

The self-healing ability means that the composite service can heal itself, if any execution problems occur, in order to successfully complete its execution, while respecting QoS agreements. QoS-driven self-healing approach can be divided into two categories: one is aiming at local QoS optimization, and the other is for global QoS optimization.

The former one adopts multi-object policy to score candidate services and, based on such scores, reselects an optimal candidate service for the service class to which the failed service belongs. For example, [12] proposes a service replication approach, in order to substitute the original component service when it is not available due to the heavy load of the network. Based on the idea of replication, [13] proposes a service composition approach based on redundant mechanism. The key to this approach is to establish a set of redundant services for each component service. Then, if one component service fails, the service can be replaced with an alternative member of the same redundancy group. In our study, in order to achieve end-to-end constraint, we attempt to research on issues of the self-healing approach with global QoS optimization.

The self-healing Web service composition with global QoS optimization adopts a global optimization policy to re-select the services in order to satisfy the end-to-end constraint. For example, based on the idea of the replacement composite service, the researchers[7,8] propose the approaches to backing up a composite service for each component service. Then, when a component service incurs a failure, the composite service can easily switch to a replacement and such a self-healing process will not cause an extra delay. In [7, 8], all the replacement composite services are backed up before the execution of the composite service. Such two approaches do not consider the QoS in the execution of the composite service. Because of the dynamic nature of Web services, the replacement may not be available sometimes. One of the studies of reselection in the execution of the composite service is the approach in [6]. In [6], the reselection will be triggered as soon as the actual QoS deviates from the initial estimates. When the failure is found, the execution of the composite service will be stopped until the reselection completes. The reselection will cost an extra delay of the composite service. Thus, this approach can only be used for runtime-unaware application. In this paper, we share the idea of backing up composite service in order to make the composite service automatically switch to the new replacement when component service fails. Comparing with the above work, we aim at improving the availability of replacement composite service through

adding a process of finding the replacement if the replacement backed before is not available. Besides, a performance prediction is used in order to reduce the extra cost caused by the reselection in the execution.

3 Motivation

3.1 Preliminaries

In this section, we will introduce the concept of QoS and the selection with global optimization to analyze the necessity of self-healing ability of composite service. According to the analysis, the problem of selection with global optimization is formulated and it can be used as a basis for reliability-oriented QoS-driven reselection.

Definition 1 (QoS of Atomic Service). For an atomic service s (which only contains one operation), the QoS of s can be defined as: $\mathrm{QoS}(s) = \langle Q^t(s), Q^p(s)\rangle$, where

• $Q^t(s)$ is the response time of s which is the interval of time elapsed from the invocation to the completion of service s. $Q^t(s)$ can be defined as: $Q^t(s) = \langle Q^{pt}(s), Q^{dt}(s)\rangle$, where

▲ $Q^{pt}(s)$ is the processing time for the request, which is provided by service providers;

▲ $Q^{dt}(s)$ is the sum of the transmission time of request and response between the execution engine and the service. The transmission time can be estimated based on past statistics and then it can be either an average of past executions of the service, or the transmission time with a given probability, e.g., 95%;

▲ $Q^t(s) = Q^{pt}(s) + Q^{dt}(s)$ means that the response time is a sum of request processing time and data transmission time;

• $Q^p(s)$ is the price of invoking s.

Cardoso[14] proposes a mathematical model for QoS computation of workflow, which involves aggregation functions for sequence, choice, parallel and iteration structure of the workflow. In this paper, we use the aggregation functions as [14]. For the purpose of providing composite services which satisfy global constraints and preferences defined by users, QoS-driven selection with global optimization is proposed in [3]. The problem can be formulated as (1). The aim of the selection is to maximize the fitness function of the QoS; and to meet the constraints defined by the user.

$$\begin{aligned} &\max \sum_{i=1}^{N} \sum_{j \in S_i} F_{ij} x_{ij} \\ &\text{s.t.} \sum_{i=1}^{N} \sum_{j \in S_i} Q_{ij}^z x_{ij} \leqslant Q_c^z, \quad z \in \{p, t\} \\ &\quad \sum_{j \in S_i} x_{ij} = 1,\ x_{ij} \in \{0, 1\},\ i = 1, \ldots, N,\ j \in S_i, \end{aligned} \tag{1}$$

where x_{ij} is set to 1, if atomic service j is selected for service class S_i in the workflow and, 0 otherwise. Q_{ij} is the QoS value of service j in class S_i. F_{ij} is a fitness function which can be computed as (2); Q_c^z is the expected QoS of the composite service.

$$F_{ij} = w_t \times \Big(\frac{Q_{ij}^t - u_t}{\sigma_t}\Big) + w_p \times \Big(\frac{Q_{ij}^p - u_p}{\sigma_p}\Big), \tag{2}$$

where w_t and w_p are the weights ($0 \leqslant w_t,\ w_p \leqslant 1$, $w_t + w_p = 1$). σ and μ are the standard deviation and average of the QoS values respectively, for all candidate services in a service class.

From (1), QoS-driven selection with global optimization is an NP hard problem[4]. Normally, it takes a long time to solve this problem. After the selection, the composite service will be executed.

3.2 Scenarios

In order to illustrate our approach, we present a Web service composition of travel planning, by which a user may create a vacation package. Fig.1 illustrates the main steps that are needed in the composition. In the composite process, a flight booking is done in parallel with a hotel reservation, after operations of booking and reservation, the distance from the airport to the hotel is computed, and then either a car rental or a bike rental is invoked.

The goal of the QoS-driven selection with global optimization is to select a candidate component service from each service class in order to make the composite service satisfy user's requirements and make the fitness

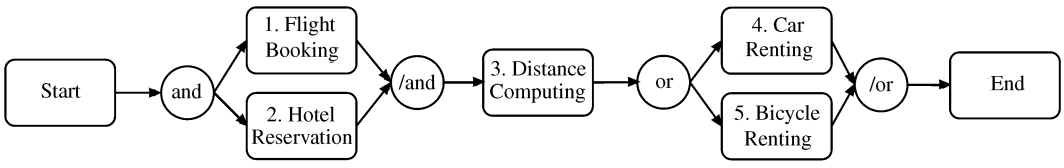


Fig.1. Composite process of travel planning.

function maximized. After the selection, composite service will run.

Due to the highly dynamic environment of Web service, the QoS of a component service in the execution may be deviated from the estimated one. In order to satisfy the QoS requirement, the composite service will heal itself by reselection.

Suppose the component service for service class of Distance Computing is invoked and the service fails. Under this situation, there is a reselection, that means to reselect the slice containing the service class of Distance Computing and its successors. Such reselection is also an NP hard problem and it may need a longer time. Meanwhile, the composite service will not run until the reselection finishes and then, the composite service will be interrupted for a period of time. So the healing approach can be used only for runtime-unawareness composition. However, if the composite service has a long response time, users may not have enough patience to wait for the response. Under this situation, users may resort to another provider. Also, since users may drop out the original composition, service providers may not gain profits from the service they provide. Therefore, the cost of extra delay needs to be reduced.

Another way of self-healing Web service composition is based on the idea of finding the replacement composite service. In this way, a replacement composite service for each component service is backed up in the selection.

Suppose $\{s_{11}, s_{21}, s_{31}, s_{41}, s_{51}\}$ is the selected composite service for the composite process illustrated in Fig.1. In the selection, a sub-optimal composite service is also backed up for the component service s_{21}, that is, $\{s_{22}, s_{32}, s_{42}, s_{52}\}$. At the invocation time of s_{21}, if s_{21} is found to fail, the composite service can automatically switch to the replacement composite service of s_{21}. Then the unexecuted part of the composite service becomes a new one $\{s_{22}, s_{32}, s_{42}, s_{52}\}$ and the composite service can heal itself to continue its execution.

However, since the replacement composite service is backed up in the selection, the QoS of component services in the replacement may change. This may make the composite service unavailable. In this situation a reselection may be needed which will interrupt the execution of the composite service and might cause an extra delay.

Performance prediction is the key to our approach that makes the composite service heal itself from the failure as quickly as possible and minimize the number of reselections. Firstly, by performance prediction, component services which will incur a QoS violation can be predicted, and if the corresponding substitute composite service is not available, the reselection aiming at finding new replacement one will be triggered as early as possible. It will be more likely to complete the reselection before the invocation of the component service and thus the composite service can repair itself from the failure as quickly as possible. Secondly, by performance prediction, the reliability of the services can be modeled and quantified, on the basis of which the QoS-driven reselection can find a replacement composite service with high reliability and better QoS. This will make the replacement composite service more reliable and minimize the number of the reselections. All in all, by performance prediction, the composite service can be more adaptive to the dynamics of Web service.

4 Performance Prediction Based on Semi-Markov Model

Web services operate autonomously within the highly changeable environment of the Web. As a result, the QoS may change frequently. The changes may be caused either by service providers, who can minimize the price for invoking the service, or by the network, whose heavy network load may affect the data transmission time. Compared with the changes caused by service providers, the changes caused by the network may occur more frequently. Changes caused by the network may affect the data transmission speed and thus, affect the response time of the composite service. Therefore, in this study, we try to predict the data transmission speed.

The study of this paper is based on the following assumptions: (a) the speed of processing the request is constant; (b) the price of requesting a service is constant; (c) the execution engine never fails; (d) the failures by different services and communication links are separate and do not interfere with each other; and (e) during the data transmission process, data transmission speed does not change.

4.1 Semi-Markov Model for Data Transmission Speed

Markov Process model is one of the probabilistic models. It is useful in analyzing dynamic behaviors of the system[15]. A Semi-Markov Process (SMP) is the extension of Markov's, which models time-dependent stochastic behaviors[16]. An SMP is similar to Markov Process except that its transition probabilities depend on the amount of time that elapses since the last transition. Since Web services operate with frequent changes, the change of data transmission speed can result either from the soft damage of the Internet, e.g., network load, or from the hard damage of the Internet like a

communication link that is broken. In this sense, the stochastic behavior of the data transmission speed depends not only on the current state, but also on the duration of the state. Thus, an SMP can be used to analyze the behavior of data transmission speed.

In order to use SMP to analyze the stochastic behavior of data transmission speed, we firstly classify data transmission speed into three states: qualified state, soft damage state and hard damage state. The meanings of the above three states are given in Definition 2, where $V(t)$ is used to denote the data transmission speed at time t and $ST(t)$ is used to denote the state of the data transmission speed at time t.

Definition 2 (States of Data Transmission Speed). *We use th_V_Q to denote the threshold of data transmission speed in the qualified state.*

- *If $V(t) \geqslant th\_V_Q$, then $ST(t) =$ qualified state.*
- *If $0 < V(t) < th\_V_Q$, then $ST(t) =$ soft damage state.*
- *If $V(t) = 0$, then $ST(t) =$ hard damage state.*

In this paper, th_V_Q is the data transmission speed with a given probability of 70% of the past executions. In soft damage state, data transmission speed is below the speed in qualified state because of the increase of network load and, after a period of time, the data transmission speed can be automatically improved. In hard damage state, data transmission speed is closer to 0, as a result of the damage of the hardware of network. In this state, only after the manual reparation of the hardware, the data transmission speed will be improved and this repairing process will take longer time.

After defining the states of data transmission speed, a semi-Markov model for data transmission speed can be defined as follows.

Definition 3 (Semi-Markov Model for Data Transmission Speed). *A Semi-Markov model for data transmission speed can be defined as: $SM = \langle Z, P, F \rangle$, where*

- *Z is the state space of data transmission speed, $Z = \{1, 2, 3\}$. When $Z = i$, it means that the data transmission speed is in state i. States 1, 2 and 3 are qualified, soft damage, and hard damage states respectively;*
- *P is the matrix of state transition probabilities. If the current state is i, the next state enters in state j with probability P_{ij} and $\sum_j P_{ij} = 1$. Especially, $P_{ii} = 0$;*
- *if the current state is i and the next state is j, the duration at state i until the completion of transition from i to j is d. d obeys the distribution $F_{ij}(d)$.*

Let $H_i(d)$ be the distribution of duration spent in state i, $H_i(d) = \sum_j F_{ij}(d) \times P_{ij}$. The average duration in state i can be signified as μ_i. According to the lemmas[17] of semi-Markov model, there is stationary distribution $\pi = [\pi_1, \pi_2, \pi_3]$, and for each π_j, it can be computed through getting the solution to (3). Also, let $LimP_i$ be the steady-state occupancy probability of state i, it can be computed as (4).

$$\begin{cases} \pi_j = \sum_{i=1}^{5} \pi_i P_{ij}, \\ \sum_{i=1}^{5} \pi_i = 1; \end{cases} \tag{3}$$

$$LimP_i = \frac{\pi_i \mu_i}{\sum_j \pi_j \mu_j}. \tag{4}$$

Fig.2 shows the semi-Markov model for data transmission speed. In Fig.2, each circle signifies a state. The arcs between circles signify the transitions. The probability of each transition is labeled on the arc.

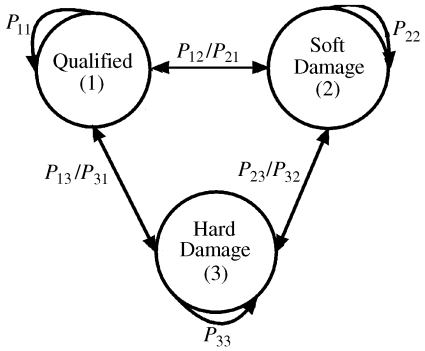


Fig.2. Semi-Markov model for data transmission speed.

4.2 Performance Prediction Based on Semi-Markov Model

In the execution of the composite service, if data transmission speed of one of the unexecuted component services is a great deviation from the estimated one, there might be a risk of violating the overall QoS of the composite service. In order to reduce the damage, the composite service must heal itself. Aiming at improving the availability of the replacement composite service and making the original composite service heal itself as quickly as possible, this study intends to predict the QoS performance.

4.2.1 Description of Prediction

As to our prediction problem, we try to predict whether the data transmission speed of the service is a deviation from the estimated one during the invocation.

Such a problem can be described as follows: if the current state is i, current time (predicted time) is t and the duration in the current state is d, we need to predict the probability of that the data transmission speed V_f at future time (the time to be predicted). t_f must not be below the maximum deviation V_e. Let j be the state of V_e. To solve this problem, we need to consider the following two situations.

Situation 1. State j is the same as i.

This situation is either because of lasting in state i from t to t_f, or as a result of switching to i again after multiple transitions. We will consider these two situations.

Situation 1(a): State j is the same as i and during $t_f - t$ no transition occurs.

Let D_i be the stochastic variable of duration in the state i. If no transition occurs during $t_f - t$, it means that the duration in state i will be $d + t_f - t$. Then, the probability of data transmission speed V_f at future time t_f that is not below the maximum speed V_e is computed as (5).

$$\begin{aligned} PR &= PR_{1a} = P\big((V \geqslant V_e) \wedge (D_i \geqslant t_f - t + d | D_i \geqslant d)\big) \\ &= P(V \geqslant V_e) \times P(D_i \geqslant t_f - t + d | D_i \geqslant d) \\ &= (1 - FV_i(V_e)) \\ &\quad \times \frac{P(D_i \geqslant d | D_i \geqslant t_f - t + d) \times P(D_i \geqslant t_f - t + d)}{P(D_i \geqslant d)} \\ &= (1 - FV_i(V_e)) \times \frac{1 - H_i(t_f - t + d)}{1 - H_i(d)}, \end{aligned} \tag{5}$$

where $FV_i(v)$ is the distribution of data transmission speed in state i; $P(D_i \geqslant d + t_f - t | D_i \geqslant d)$ is the probability of duration in state i bigger than $d + t_f - t$, under the condition that it has been already in state i for d since the last transition.

Situation 1(b). State j is the same as i and during $t_f - t$, at least one transition occurs.

If at least one transition occurs, it means that the duration kept in state i before the first transition is shorter than $d + t_f - t$. Meantime, the probability of re-entering state i after several transitions during the interval from t_f to t is close to the steady-state occupancy probability $LimP_i$ of state i. The probability can be computed as (6).

$$\begin{aligned} PR &= PR_{1b} = P\big((V \geqslant V_e) \wedge (Z_{t_f} = i) \\ &\qquad \wedge (d \leqslant D_i < d + t_f - t | D_i \geqslant d)\big) \\ &= P(V \geqslant V_e) \times P(Z_{t_f} = i) \\ &\qquad \times P(d \leqslant D_i < d + t_f - t | D_i \geqslant d) \\ &= (1 - FV_i(V_e)) \times LimP_i \\ &\quad \times \frac{1}{P(D_i \geqslant d)} \big\{ P(D_i \geqslant d | d \leqslant D_i < t_f - t + d) \\ &\quad \times P(d \leqslant D_i < t_f - t + d) \big\}^{-1} \\ &= (1 - FV_i(V_e)) \times LimP_i \times \frac{H_i(t_f - t + d) - H_i(d)}{1 - H_i(d)}, \end{aligned} \tag{6}$$

where $P(d \leqslant D_i < d + t_f - t | D_i \geqslant d)$ is the probability being spent in state i with the duration not less than d and more than $d + t_f - t$; $P(Z_{tf} = i)$ is the probability of the transition entering state i from the other state.

Considering the above two situations, the probability under the situation that state j is similar to i can be computed as (7).

$$\begin{aligned} PR_1 &= P((V \geqslant V_e) \wedge (D_i \geqslant d)) = PR_{1a} + PR_{1b} \\ &= (1 - FV_i(V_e)) \times \frac{1 - H_i(t_f - t + d)}{1 - H_i(d)} \\ &\quad + (1 - FV_i(V_e)) \times LimP_i \\ &\quad \times \frac{H_i(t_f - t + d) - H_i(d)}{1 - H_i(d)}. \end{aligned} \tag{7}$$

Situation 2. State j is different from i.

If state j is different from i, it means that there exists at least one transition during the interval from t to t_f. Similar to *Situation* 1(b), the duration kept in state i before the first transition is shorter than $d + t_f - t$. Meanwhile, the probability of entering state j after several transitions during the interval from t_f to t is close to the steady-state occupancy probability $LimP_j$ of state j. Then, the probability can be computed as (8).

$$\begin{aligned} PR &= PR_2 = P((V \geqslant V_e) \wedge (Z_{t_f} = j) \\ &\qquad \wedge (d \leqslant D_i < d + t_f - t | D_i \geqslant d)) \\ &= P(V \geqslant V_e) \times P(Z_{t_f} = j) \\ &\qquad \times P(d \leqslant D_i < d + t_f - t | D_i \geqslant d) \\ &= (1 - FV_j(V_e)) \times LimP_j \\ &\qquad \times \frac{H_i(t_f - t + d) - H_i(d)}{1 - H_i(d)}, \end{aligned} \tag{8}$$

where $P(Z_{tf} = j)$ is the probability of the transition entering state j from the other state.

Considering the above two situations, the probability of the data transmission speed V at time t_f that is not below the maximum deviation speed V_e can be computed as (9). When the probability is bigger, it means that the data transmission speed V is most likely bigger than the deviation speed. Thus, the probability can reflect whether the QoS of the component the services in its invocation will be violated or not, that is, the

reliability of component services can be reflected.

$$PR = \begin{cases} PR_1, & i = j \\ PR_2, & i \neq j. \end{cases} \tag{9}$$

4.2.2 Process of Prediction

In order to predict the data transmission speed, we need to get the distribution of data transmission speed $F_i(V)$ in state i, the distribution of duration $F_{ij}(d)$ and transition probability P_{ij}. For the purpose of calculating the above distributions and probability, for a continuous-time semi-Markov process, a set of backward Kolmogorov integral equations[16] are developed. While the approaches to these equations are feasible and can achieve accurate results in some situations, they perform poorly in many other situations, for instance, when the rate of transitions is exponential with time. In the applications like [17], a discrete-time semi-Markov process is often utilized to achieve simplification. In this study, we develop a discrete-time semi-Markov process to calculate the distributions and probability.

In the discrete-time semi-Markov model, the distribution of a discrete-time variable y depends on the past statistics. Then, in order to get the distribution of discrete-time variable, we give the definition of QoS-related contexts.

Definition 4 (QoS-Related Contexts). *The QoS-related context is the observed result at time t_{ob}, and it can be defined as $qc = \langle t_{ob}, v, st_{ob} \rangle$, where t_{ob} is the time of the observation; v is the observed data transmission speed at t_{ob}; st_{ob} is the state of data transmission speed v.*

In the following part, we will introduce how to compute the needed probability and distributions based on the QoS-related contexts.

• Establishing Transition Set

Let $QCS = \{qc_1, qc_2, \ldots, qc_n\}$ be a set of QoS-related contexts observed in the past. For each $qc_i \in QCS$, qc_i is the QoS-related context observed at time $qc_i.t_{\text{ob}}$.

Given QCS, all the transitions from state i to j form a set $TR_{ij} = \{(qc_u, qc_v, duration) | qc_u,\ qc_v \in QCS \wedge qc_u.st_{\text{ob}} = i \wedge qc_v.st_{\text{ob}} = j \wedge i \neq j\}$. In this study, we use $TR_{ij}[k]$ to signify an element $(qc_u, qc_v, duration)$ in TR_{ij} where $duration$ is the time between $qc_u.t_{\text{ob}}$ and $qc_v.t_{\text{ob}}$. And $duration$ can be computed as (10). We use $TR_{ij}[k].pre$ to signify qc_u and $TR_{ij}[k].post$ to signify qc_v.

$$TR_{ij}[k].duration = TR_{ij}[k].post.t_{\text{ob}} - QCS[l].t_{\text{ob}}, \tag{10}$$

where $QCS[l]$ is an element in QCS and it satisfies the following conditions: $QCS[l-1].st_{\text{ob}} \neq TR_{ij}[k].pre.st_{\text{ob}}$ and for any qc in QCS, if $QCS[l].t_{\text{ob}} \leqslant qc.t_{\text{ob}} < TR_{ij}[k].pre.t_{\text{ob}}$, $qc.st_{\text{ob}} = TR_{ij}[i].pre.st_{\text{ob}}$.

• Calculating Transition Probability P_{ij}

Based on the transition sets TR, the probability of transition from state i to j can be computed as

$$P_{ij} = \frac{|TR_{ij}|}{\sum_k |TR_{ik}|}, \tag{11}$$

where TR_{ij} is the set of transitions from state i to j according to QCS; $|TR_{ij}|$ is the number of elements in TR_{ij}.

• Computing Steady-State Occupancy Probability $LimP_i$

To compute the steady-state occupancy probability, we firstly get the solution to (3), to obtain π. As (3) can be easily transformed to (12), π can be computed as

$$[\pi_1\ \pi_2\ \pi_3] \bullet \begin{bmatrix} 1 & -P_{12} & 1 \\ -P_{21} & 1 & 1 \\ -P_{31} & -P_{32} & 1 \end{bmatrix} = [0\ 0\ 1], \tag{12}$$

$$[\pi_1\ \pi_2\ \pi_3] = [0\ 0\ 1] \bullet \begin{bmatrix} 1 & -P_{12} & 1 \\ -P_{21} & 1 & 1 \\ -P_{31} & -P_{32} & 1 \end{bmatrix}^{-1}. \tag{13}$$

Let $\boldsymbol{u} = [u_1,\ u_2, \ldots, u_5]$. For any u_i, u_i is the average duration spent in state i, and it can be computed as

$$u_i = \frac{\sum_j \sum_k TR_{ij}[k].duration}{\sum_j |\{TR_{ij}[k]\}|}. \tag{14}$$

After getting π and u, by (4), the steady-state probability of each state can be computed.

• Getting Distribution of Duration $H_i(d)$

To compute the probability as in (9), what is needed is the duration spent in the current state since last transition. Let TR_{final} be the final transition according to QCS, and $TR_{\text{final}}.post$ be qc. Then, the state of qc is the current state. Let the current time be t. Then, the duration is $t - qc.t_{\text{ob}}$.

Let us discuss how to compute $H_i(d)$. We will compute $F_{ij}(d)$ firstly. $F_{ij}(d)$ is the probability of entering the state j from state i, under the condition that the duration in state i is less than d. Then, $F_{ij}(d)$ will be the ratio of the number of elements in TR_{ij} with the duration that is below d to the number of all elements in TR_{ij}. It can be computed as (15). Then, $H_i(d)$ can

be computed as (16).

$$F_{ij}(d) = P\{T < d\} = \frac{|\{TR_{ij}[u]|u \in [1, |TR_{ij}|] \wedge TR_{ij}[u].duration < d\}|}{|\{TR_{ij}[u]\}|} \quad (15)$$

$$H_i(d) = \sum_j F_{ij}(d) \times P_{ij}, \quad (16)$$

- Getting Distribution of Data Transmission Speed $FV_i(v)$

To compute the probability as in (9), it is also needed to compute the probability $FV_i(V)$ of data transmission speed that is less than the maximum deviation speed V in state i. It can be computed as

$$P(v < V \wedge Z = i) = FV_i(V) = \frac{|\{qc|qc \in QCS \wedge qc.st_{\text{ob}} = i \wedge qc.v < V\}|}{|\{qc|qc \in QCS \wedge qc.st_{\text{ob}} = i\}|}. \quad (17)$$

After computing all the above probabilities, we can use (9) to compute the probability of data transmission speed not below the maximum deviation speed. The duration in the current state since last transition will be different in different prediction processes, and thus the probability of duration in some states will be computed during the prediction process. Meanwhile, the service may be involved in different applications and it may have different maximum deviation speeds in different applications. Thus, the probability of data transmission speed below the deviation will be computed during the prediction process. Considering the transition probability P_{ij} and the steady-state probability $LimP_i$, they are relatively steady. The statistical results of them calculated before can be used in the process of prediction. In order to achieve the accuracy of the prediction, it is only needed to update $LimP_i$ and P_{ij} periodically.

Based on the above analysis, this paper presents the algorithm for determining whether a component service will incur a QoS violation or not.

Algorithm 1. Performance Prediction Based on Discrete-Time Semi-Markov Model

P Prediction$(S, T, EV, st)//S$ is the service needed to predict; T is the earliest start time of S; EV is the maximum transmission speed; st is the state of EV.

```
1  begin
2  compute the needed former information QCS;
3  establish the transition set TR from QCS;
4  get existing stationary distribution π;
5  get existing average duration spent in each state u;
6  get existing set {P_ij} of transition probabilities;
7  get existing set {LimP_i} of steady-state probabilities;
8  compute the latest transition TR_recent from TR;
9  currentState = TR_recent.st_ob;
10 lastDuration = TR_recent.duration;
11 using QCS and under st as well as EV, according to
   (17), compute FV[st][EV];
12 using TR, according to (15), compute F[current − State][][lastDuration] and F[currentState][][T−
   lastDuration] respectively;
13 according to (16), compute H[currentState] [lastDuration] and H[currentState][T-lastDuration] respectively;
14 if currentState = st then
15     compute P according to (7);
16 else
17     compute P according to (8);
18 return P;
19 end
```

If the scale of QCS is n, and the scale of $TR[i][]$ is m, the time complexity of the algorithm is $O(3m + n)$. As the time complexity is polynomial, the efficiency of the algorithm can be preserved. Then, it can react to the QoS violation quickly.

5 Self-Healing Web Service-Composition-Based Performance Prediction

5.1 Approach Overview

In this paper, we propose an approach to the self-healing of composite services when there are execution problems. Fig.3 presents the framework of the approach.

Firstly, the monitor of the QoS-related context will collect QoS-related contexts for prediction.

Secondly, QoS of each component service in the composite one will be predicted periodically. If the QoS at invocating time of certain service is predicted to be a large deviation, and the pre-backup replacement is not available, the process of reselection will be triggered.

After that, taking the original replacement composite service as a reselected slice, a reliability-oriented QoS-driven reselection will be done to get a new available replacement.

Finally, when the failure component service is to be invoked, the composite service will automatically switch to the replacement to heal itself.

The key to our approach is performance prediction. If the performance of the service is predicted to be a failure and the pre-backup replacement is not available, the process of reselection will be triggered. Compared with the work of triggering the reselection when to invoke the failure service, the approach in this paper can

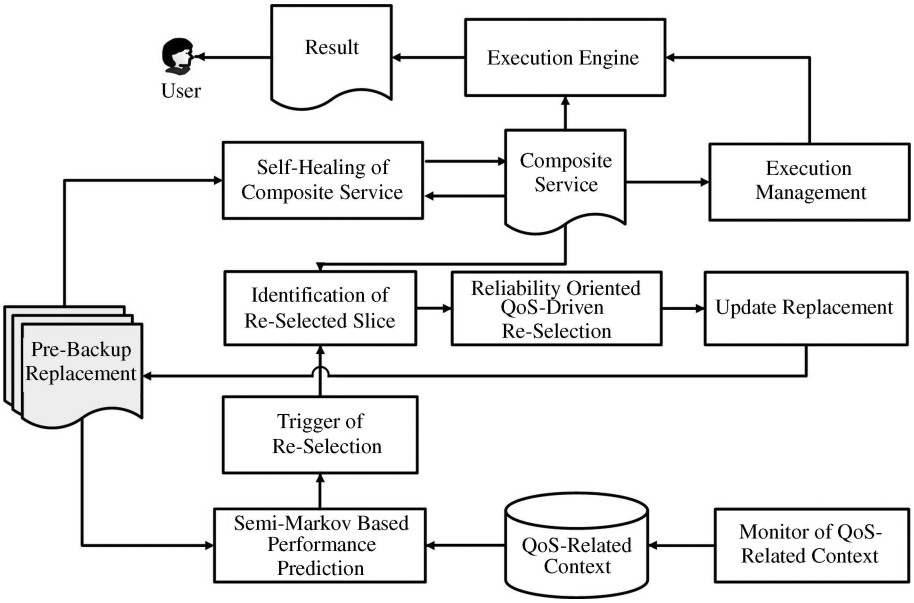


Fig.3. Performance-prediction-based QoS-aware self-healing Web service composition framework.

make the reselection process before the invocation of the failure component service. Thus, the composite service will not need to wait for the completeness of the reselection and can heal itself from the failure as quickly as possible. Meantime, the performance prediction can support the reliability modeling. Then, the reliability-oriented QoS-driven reselection will maximize the QoS of the reselected composite service and respect the reliability as well. Thus, it can minimize the number of reselections and make the composite service more adaptive.

The management of the execution of a composite service will be needed in the process of the above steps in order to interrupt and restart the execution if necessary. In this paper, we aim at the part of self-healing. The management of the execution will not be dealt with in detail.

5.2 Modeling Reliability of Web Service

QoS reflects how a service can implement functions with certain performance. In the context of composition, service reliability presents to what degree the service can meet the request according to the estimated QoS.

In the context of composition, the earliest start time et_{ij} of service j for service class i can be computed as (18).

$$et_{ij} = \begin{cases} \max\limits_{k<i}\{et_{ku} + t_{ku}\}, & k \neq 0; \\ 0, & k = 0; \end{cases} \tag{18}$$

where service class k ($k < i$) is the one that will be invoked before i and t_{ku} is the response time of service u selected for service class k; t_{ku} is the response time of service u.

Definition 5 (Reliability of Web Service). *Reliability of service s in composite service CS is the probability that a request is correctly responded with the estimated QoS. It can be defined as follows*: $R(s, CS) = \langle R_p(s, CS), R_t(s, CS)\rangle$, *where*

• $R_p(s, CS)$ *is the request processing reliability, which can be defined as*

$$R_p(s, CS) = e^{-\lambda t}, \tag{19}$$

where λ is the failure rate of service when processing a request; t is the sum of response time and the earliest start time of s in CS; (19) *is a common part in software or hardware components' reliability, which has been justified in both theory and practice*[11].

• $R_t(s, CS)$ *is the data transmission reliability. According to the performance prediction, the transmission reliability of service s in the context of composition can be computed as*

$$R_t(s, CS) = P, \tag{20}$$

where P can be computed as (9).

• *The overall reliability of service s can be computed as*

$$R(s, CS) = R_p(s, CS) \times R_t(s, CS). \tag{21}$$

For a composite service, only when all the component services are reliable, the composite one can be reliable. So the reliability of the composite service is the multiplication of its component ones.

Definition 6 (Reliability of Composite Service). *Reliability of composite service CS is the probability of all the component services that execute as the estimated QoS, which can be defined as*

$$R(CS) = \prod_{v} R(s_v, CS), \tag{22}$$

where s_v is a component service of CS.

Then, Algorithm 2 is presented to compute the reliability of composite service based on the proposed algorithm of performance prediction. Algorithm 3 is to compute the reliability of service.

Algorithm 2. Computing Reliability of Composite Service

Reliability RCS(*CS*)//*CS* is the composite service

```
1  begin
2  for each s in CS do
3  begin
4      get the reliability Rs of s in context of CS as
       Algorithm 3;
5      R = R × Rs;
6  end
7  return R;
8  end
```

Algorithm 3. Computing Reliability of Service

Reliability RS(*CS*, *s*)

```
1  begin
2  Get QoS property cq, where cq.f is the failure rate
   of processing request; cq.tp is the request process-
   ing time announced by providers; cq.c is the cost
   for invoking service; cq.v is the maximum data
   transmission speed;
3  get the earliest start time et of s in context of CS;
4  compute processing reliability Rp according to
   (19);
5  compute transmission reliability Rt as Algorithm 1;
6  return Rp × Rt;
7  end
```

5.3 Triggering Reselection Based on Performance Prediction

Algorithm 4 describes the proposed reselection triggering method. The basic idea is to predict the performance of each component service in the composite one, and when the predicted QoS is below the threshold (which means the service will not be reliable or a failure will occur at invocation time) and no replacement can be used to rescue such failure, the reselection will be triggered.

Algorithm 4. Reselection Triggering Algorithm

service TriggerReselection(*CS*, *ReplaceCS*)//*CS* is the composite service.

```
1  begin
2  for each service s in CS do
3  begin
4      compute reliability Rs of s as Algorithm 3;
5      if Rs < Threshold then
6      begin
7          compute reliability R of original replace-
           ment composite service ReplaceCS[s] of s
           as Algorithm 2;
8          if R < Threshold_CS then //if replace-
             ment one is not available
9            return s;
10     end
11 end
12 end
```

5.4 Reliability-Oriented QoS-Driven Reselection

The original replacement is not available and may affect the self-healing ability of a composite service. Thus, a reselection is needed to find a new replacement. Such a replacement has the same service classes as the original one. Then, the reselected slice of the composite service is the same as the corresponding workflow of the original pre-backup composite service. And the reselection will be done on the original pre-backup service. To improve the QoS of a composite service while achieving the reliability, we proposes a reliability-oriented QoS-driven reselection. The problem of such reselection can be described as

$$\begin{aligned}
&\max \sum_{i=1}^{n} \sum_{j \in S_i} F_{ij} x_{ij}, \\
&\text{s.t.} \sum_{i=1}^{n} \sum_{j \in S_i} Q_{ij}^{z} x_{ij} \leqslant Q_c^{z\prime}, \quad z \in \{p, t\}, \\
&\prod_{i=1}^{n} \prod_{j \in S_i} (Q_{ij}^{R})^{x_{ij}} \geqslant R_c, \\
&\sum_{j \in S_i} x_{ij} = 1, \quad x_{ij} \in \{0, 1\},\ i = 1, \ldots, n,\ j \in S_i,
\end{aligned} \tag{23}$$

where x_{ij} is set to 1 if atomic service j is selected for service class S_i in the workflow of the original replacement composite service, and 0 otherwise. $Q_c^{z\prime}$ is the QoS of original reselected slice which can be used as constraint for reselection; R_c is the reliability constraint.

Such a problem is a multi-constraint satisfaction problem, which can be solved by an integer programming algorithm[3] for the complex composite process, or a dynamic programming algorithm[18] for

the pipeline composite service. And lots of artificial intelligent algorithms, such as genetic algorithm[19] and simulated annealing algorithm[20], are proposed to solve this problem. As the limitation of this paper, we will not discuss how to solve this problem in detail.

6 Experiments

This part will verify the effectiveness of the proposed approach.

Experiment 1 is used to test the effectiveness of the proposed performance predicting approach based on the semi-Markov model. We simulate test set of data transmission speed according to the Gaussian distribution $N(10, 1)$. The threshold of reliability is 0.9. The size of test set is 100 000. We compare the relations among the predicted results, predicted interval and the observed interval between two neighboring contexts. Table 1 gives the results (O_{Q} is the observed interval between two neighboring QoS-related contexts in the test set; N is the number of predictions; R is the average accurate rate of the predictions which can be computed as the ratio of the number of predictions that is right to the number of predictions; I is the predicted interval and the unit of I is second).

Table 1 shows that if the observed interval between two neighboring contexts is smaller than 0.1s, although the predicting interval is relatively bigger (e.g., 60s), the predicted result (accuracy is 93%) is acceptable. If the observed interval is smaller (e.g., 0.05s), although the predicted interval is bigger (e.g., 180s), the accuracy of the predicted result (e.g., 92%) is also acceptable. Thus, through minimizing the observation interval, the accuracy of prediction result can be improved.

Experiment 2 is to test the self-healing ability of the approach. 10 scenarios with $O_{\mathrm{Q}} = 0.1$s, $I = 60$s are randomly generated, and the threshold of reliability is 0.9. A comparison of the extra delay by the following three approaches is presented in Fig.4.

Fig.4 shows that the extra delay caused by the proposed approach is always the shortest among the three approaches. The reason is that the proposed approach is an improvement of the traditional pre-backup approach. By performance prediction, component services which will incur a QoS violation can be predicted and if the corresponding replacement composite service is not available, the reselection process aiming at finding a replacement will be more likely to finish before the invocation of the component service and thus the composite service can heal itself from the failure as quickly as possible. Then, the extra cost caused by such a self-healing process will be the minimum and the proposed approach will make the composite service heal itself much more quickly.

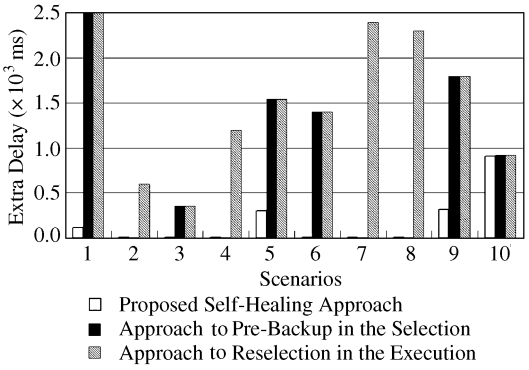


Fig.4. Comparison of extra delay.

Experiment 3 is used to test the availability of a replacement composite service. 10 composite services are randomly generated, for each composite service, we simulate 10 failure situations and set $O_{\mathrm{Q}} = 0.1$s, $I = 60$s, the threshold of reliability is 0.9. We compare the success rate of substitution before invoking time of the failure component service. The result is shown in Fig.5.

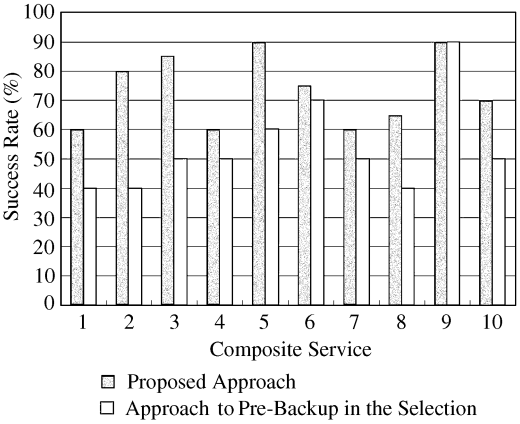


Fig.5. Comparison of success rates.

Table 1. Semi-Markov Based Predicted Results

O_{Q}	0.5s			0.1s			0.05s		
I	10	60	180	10	60	180	10	60	180
N	300	300	300	200	200	200	150	150	150
R%	95	86	80	97	93	83	98	95	92

Fig.5 shows that the success rate of the proposed approach is always better than that of traditional pre-backup approach. The reasons are as follows. First, the proposed one considers the QoS performance of component services during the execution of the composite service, the availability of the replacement composite service can be achieved. Thus, the success rate will

be high. Second, during the reselection process, both QoS of the composite service and the reliability of composite service are emphasized. Thus, the more reliable replacement composite service will minimize the number of reselection and improve the success rate of the substitution.

The above experiments show that the proposed approach is more effective in supporting the composite service to adapt to the dynamics of Web service.

7 Conclusion

We propose a self-healing approach in order to make composite service adapt to the dynamic property of Web services. Such an approach is an integration of backing up in the selection and reselecting in the execution. A way of performance prediction is proposed to make the composite service heal itself as quickly as possible and minimize the number of reselections. On this basis, the self-healing approach based on performance prediction is presented including the framework of this approach, the triggering algorithm of the reselection and the reliability model of the service.

In the future work, we will study on how to adapt the approach to deal with more complicated and realistic scenarios, where data transmission speed, failure rate and the cost of requesting a service are not assumed to be constant.

References

[1] Milanovic N, Malek M. Current solutions for Web service composition. *IEEE Internet Computing*, 2004, 8(5): 51–59.

[2] Zhang L J, Li H F, Lam H. Services computing: Grid applications for today. *IT Professional*, 2004, 6(4): 5–7.

[3] Zeng L Z, Benatallah B. QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering*, 2004, 30(5): 311–327.

[4] Yu T, Zhang Y, Lin K J. Efficient algorithms for Web services selection with end-to-end QoS constraints. *ACM Transactions on the Web*, 2007, 1(1): Article 6.

[5] Bonatti P A, Festa P. On optimal service selection. In *Proc. Int. Conf. World Wide Web*, Chiba, Japan, May 2005, pp.530–538.

[6] Canfora G, Penta M D, Esposito R *et al*. QoS-aware replanning of composite web services. In *Proc. Int. Conf. Web Services*, Orlando, USA, July 2005, pp.121–129.

[7] Yu T, Lin K J. Adaptive algorithms for finding replacement services in autonomic distributed business processes. In *Proc. International Symposium on Autonomous Decentralized Systems*, Chengdu, China, April 2005, pp.427–434.

[8] Girish C, Koustuv D, Arun K *et al*. Adaptation in Web service composition and execution. In *Proc. Int. Conf. Web Services*, Chicago, USA, September 2006, pp.549–557.

[9] Huang G, Zhou L, Liu X Z *et al*. Performance aware service pool in dependable service oriented architecture. *Journal of Computer Science and Technology*, 2006, 21(4): 565–573.

[10] Dai Y S, Levitin G, Trivedi K S. Performance and reliability of tree-structured grid services considering data dependence and failure correlation. *IEEE Trans. Comput.*, 2007, 56(7): 925–936.

[11] Xie M, Dai Y S, Poh K L. Computing Systems Reliability: Models and Analysis. Kluwer Academic, 2004.

[12] Jorge S, Francisco P S, Marta P M *et al*. WS-replication: A framework for highly available Web services. In *Proc. Int. Conf. World Wide Web*, Edinburgh, Scotland, May 2006, pp.357–366.

[13] Guo H P, Huai J P, Li H *et al*. ANGEL: Optimal configuration for high available service composition. In *Proc. Int. Conf. Web Services*, Salt Lake City, USA, July 2007, pp.280–287.

[14] Cardoso J, Sheth A P, Miller J A *et al*. Quality of service for workflows and Web service processes. *Journal of Web Semantics*, 2004, 1(3): 281–308.

[15] Malhotra M, Reibman A. Selecting and implementing phase approximations for semi-Markov models. *Communication Statistics-Stochastic Models*, 1993, 9(4): 473–506.

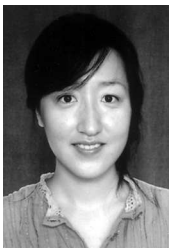
[16] Altinok Y, Kolcak D. An application of the semi-Markov model for earthquake occurrences in North Anatolia, Turkey. *Journal of the Balkan Geophysical Society*, 1999, 2(4): 90–99.

[17] Fang Z B, Miao B Q. Stochastic Process. University of Science and Technology of China Press, Hefei, 2007.

[18] Yu T, Lin K J. Service selection algorithms for Web services with end-to-end QoS constraints. In *Proc. Int. Conf. E-Commerce Technology*, California, USA, July 2004, pp.129–136.

[19] Cardoso J, Sheth A P, Miller J A *et al*. Modeling quality of service for workflows and Web service processes. *Journal of Web Semantics*, 2004, 1(3): 281–308.

[20] Jin H, Chen H H, Chen J *et al*. Real-time strategy and practice in service grid. In *Proc. Annual International Computer Software and Applications Conference*, Hong Kong, China, January 2004, pp.161–166.



Yu Dai receives her Ph.D. degree from the College of Information Science and Technology at Northeastern University, Shenyang, China in 2008. She is a lecturer of college of software, Northeastern University. Her main research interests include service composition and service oriented computing.

Lei Yang is a lecturer in the College of Information Science and Technology at Northeastern University, Shenyang, China. He received his Ph.D. degree from Northeastern University in 2007. His research interests include service composition and service oriented computing.

Bin Zhang is a professor in the College of Information Science and Technology at Northeastern University, Shenyang, China. He is a member of CCF. He received his Ph.D. degree from Northeastern University in 1997. His current research interests are service oriented computing and information retrieval.