Peer-to-Peer Networking and Applications (2019) 12:502–513
https://doi.org/10.1007/s12083-017-0625-x

Behavior Reconstruction Models for Large-scale Network Service Systems

Zhaohui Zhang [1,2,3] · Lina Ge [4] · Pengwei Wang [1,2,3] · Xinxin Zhou [1]

Received: 27 August 2017 / Accepted: 24 November 2017 / Published online: 2 January 2018

Abstract
In large-scale network service systems, the phenomenon of instantaneous gathering of a large number of users can cause system abnormality, whenever the load imposed by the user behaviors does not match the system load. This paper proposes a behavior reconstruction model for large-scale network service systems integrated with Petri net reconstruction methodology, for the purpose of achieving load balancing in the system under increasing number of users. Based on the features of the user interaction behavior sequence, the behavioral load balancing model defines a user behavior membership function. Then, a random fuzzy Petri net with delay is presented to control the user behavior reconstruction. Experiments conducted by considering various changes in the number of user behaviors and their distribution in unit time demonstrate that the proposed methodology can effectively trigger the reconstructed model to balance the system load when the system load exceeds the defined warning point.

Keywords Large-scale network service system · Behavior membership function · Load balancing · Behavior reconstruction · Petri net

1 Introduction

In the recent years, large-scale service systems based on Internet have witnessed rapid evolution such as growth of users, diversification of user requirements, and the openness of system services. Owing to the recent increase in the number of users using internet services, sharp gathering of users in a quick time often lead to the service unavailability issue. This is due to the fact that the new user load suddenly overloading the system, and often this imposed load surge up to paralyzing the system due to the increasing number of users. For instance, ticket booking system is often seasonal and can paralyze the system lead during peak time with the upsurge of user groups.

Large-scale user concurrent processing system are usually affected whilst expanding resources, whereby risking overloading the system due to uncertain user behaviors after expanding the computing resources. To this end, software self-adaptation strategies have been put forward to cope with this system overloading issue whilst expanding the system resources and to combat the complexities faced due to the increasing Internet service systems. It is obvious that the Internet system cannot suddenly scale to match the user behaviors. Therefore, it is important that special attention should be given to restructure the system behaviors in accordance with the changes in user behaviors by balancing the system load. Existing system load balancing methods [1–3] are mainly based on resource allocation and task scheduling strategies,

This article is part of the Topical Collection: *Special Issue on Software Defined Networking: Trends, Challenges and Prospective Smart Solutions*
Guest Editors: Ahmed E. Kamal, Liangxiu Han, Sohail Jabbar, and Liu Lu

✉ Zhaohui Zhang
zhzhang@dhu.edu.cn

Lina Ge
glngelina@163.com

Pengwei Wang
wangpengwei@dhu.edu.cn

Xinxin Zhou
158452693@qq.com

[1] School of Computer Science and Technology, Dong Hua University, Shanghai, China

[2] The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China

[3] Shanghai Engineering Research Center of Network Information Services, Shanghai, China

[4] Department of Computer Science and Technology, Anhui Normal University, Wuhu, China

but not consider when and how to dynamically reconstruct the system behaviors for the real-time load equilibrium.

Two important components should be considered for adaptive refactoring of system behavior. Firstly, classification of users according to the user behavioral characteristics and secondly, constructing the behavior flow for each user group to dynamically control the system load.

The remainder of the paper is organized as follows: Section 2 reviews the related works and Section 3 presents the proposed system behavior reconstruction model. Section 4 details the proposed Petri net model and algorithm for implementing system behavior reconstruction. Section 5 is covered with the experiments and discussion on the obtained results and Section 6 concludes this paper.

2 Related work

Recently, several research works have focused on dynamic system load balancing based on system behavior adaptive reconstruction. Doukha [4] proposed a load balancing method that distributes the beacon and fairly transmits the system load. Hwang [5] proposed the use of hardware indicators, CPU utilization and the number of online connections as a load evaluation criteria. Duan [1] used the CPU utilization rate, disk utilization ratio, page error number, request number, request response time and other relevant indicators to calculate the real-time load of the server. Gang [2] proposed a method to classify the user requested services to allocate the system resources, so as to achieve dynamic load balancing. Shailesh [3] used a fuzzy dynamic load balancing algorithm to achieve load balancing through task scheduling. Liu [6] proposed a distributed load balancing algorithm using a defined protocol sequence, and developed a model of queuing distributed asynchronous multi-server system. In order to achieve dynamic load balancing based on data stream level, Wang [7] proposed a cloud center dynamic load balancing method based on SDN (Software Defined Networks). However, these works have not given enough importance to the system behaviors and behavior time, both should be considered as essential criteria to achieve system load balancing in dynamic environments based on behavior reconstruction.

Adaptive reconstruction strategies have been the focus of a few research works. Slim [8] put forward adaptation as a key requirement for many software systems, whereby the system should be able to adapt its structure and behavior during runtime in order to respond to the changes witnessed in the operating environment and user needs. Zhang [9] proposed a new coordination method based on a reconfigurable network-event system. Pamela [10] proposed the implementation of a distributed persistence management model for reconfigurable multiprocessor systems on dynamically reconfigurable circuits. Rui [11] further proposed a dynamic adaptive wiping mechanism and Yang [12] proposed a reconfigurable architecture model based on layered hypergraph. Mohamed [13] proposed a reconfigurable and replaceable system for embedded control systems, and modeled it using Petri net. However, such research works have not considered the user requirements into account for system reconstruction. When a large number of users gather in a short time, system reconstruction may not be efficient without considering the user needs.

From the perspectives of the system behavior, Wang [14] pointed out that it is vitally important to understand user behaviors in online services and further proposed an unsupervised system based on the click traffic to check the modes of user behaviors. Luo [15] used fuzzy Petri nets to represent the fuzzy production rules, and performed a state analysis of power systems by an iterative computation of matrices. Kotevski [16] conjointly used queuing networks and Fluid Stochastic Petri Nets, and developed several performance models to analyze the behavior of complex systems. Lu [17] used a new hybrid model to explore the impacts and guidance of user behaviors on mobile banking services. Matthew [18] highlights the importance of finding out the user behaviors and using the same as the source of information by studying a long query log. Jose [19] proposed a genetic algorithm for user behavior modeling and classification from event sequences. In summary, the dynamic relationship between user behaviors and system service is vitally important and should be considered as an essential criteria whilst attempting to improve the overall system performance in balancing the system load.

To sum up, despite a number of works focused on adaptive dynamic balancing of system load, system reconstruction is hardly been considered in the state-of-the-art works to date. Reconstructing the system behavior flow to dynamically balance the system load based on user behavior characteristics can achieve effective load balancing performance in the large-scale network service systems. In this paper, the behavior reconstruction method has been exploited to balance the system load when a large number of users gather in a short time, for the purpose of achieving real-time system load balancing to maximize the processing capacity of the system. Users are classified based on their behavioral characteristics and corresponding behavior processes are constructed. The proposed reconstruction model is triggered when the system load exceeds the warning point during runtime, ultimately to balance the system load by controlling the interaction time of various types of users.

3 Model of system behavior reconstruction based on user behavior classification

Under normal conditions, large-scale network service systems can provide users with a stable and good services. But sometimes, due to the rapid expansion of the user population within a short time the system behaviors and the user behaviors may become incompatible. Thus, the system will become abnormal or even paralyzed. Now, many large scale network service systems usually continue to provide the same services to the users as before. As a result, when the user population increases rapidly, the system load will increase beyond the capacity of the system. In this scenario, the system will be overloaded and the system resources will be limited. To this end, this paper considers reducing the system load by dividing the user behaviors into different groups according to the user interaction sequence features, and by delaying the user group interaction behavior time.

Definition 1 User behaviors membership function $\mu_{U_i}(j)$. It indicates the degree of the user behavior U_i belonging to each class of S_j. $\mu_{U_i}(j)$ is defined as follow:

$$\mu_{U_i}(j) = \sqrt{\sum_{k=1}^{m} \left(u_{ik} - s_j\right)^2}, (j = 1, 2, \ldots, p),$$

where $U_i = \{u_1, u_2, \ldots, u_m\}$ $(i = 1, 2, \ldots, m)$ represents the interaction behavior sequence with the characteristics of user behavior time, assuming that the user behaviors are divided into p user groups based on the length of the interaction behaviors time; $S_j = \{s_1, s_2, \ldots, s_p\} (p \geq 1)$ represents the standard for each class of user groups.

Definition 2 User behaviors subordinate standards $d(u_i, s_j)$. It is the standard of user behaviors belonging to a specific user group, that is, $d(u_i, s_j) = \min(\mu_{U_i}(j))$. It indicates that the user behavior membership function value is kept to a minimum if the user behavior belongs to the group. Suppose that N_d $(N_d \in N^+)$ represents the number of $\mu_{U_i}(j) = d(u_i, s_j)$. Then, when $N_d = 1$, the behavior U_i belongs to the class of j user behavior group; when $N_d > 1$, the behavior U_i is randomly assigned to any kind of behavior group in the N_d classes.

Definition 3 At time t, the total number of user behaviors B_t submitted in the system is equal to the number of users, that is, $B_t = U_t$, where U_t represents the number of users in the system.

Definition 4 The real-time load L_t at time t. It is the system load corresponding to the total number of behaviors submitted by the users at time t, $L_t = B_t \times l$, where $l (l \geq 1)$ represents a system load required by a user to submit a request behavior.

Definition 5 System good service status. It is the service state when the system can provide services normally. When $0 \leq L_t \leq L_{safe}$, the system is in a good service state, where L_{safe} is the safe load, indicating that the system is in a good service state which can withstand the maximum service capacity corresponding to the load value.

Definition 6 System unstable service status. It is the service state when the system can provide services, but there may be abnormality. That is, when $L_{safe} < L_t \leq L_{\max}$, the system is in an unstable service state, where L_{max} represents the load value corresponding to the maximum service capacity that the system can withstand in the unstable service state, which is the maximum load that the system can resist.

Definition 7 System non-service status. It is the service state when the system cannot provide services because the load is too large to handle. That is, when $L_t > L_{max}$, the system is in a non-service state or in a state of paralysis.

Definition 8 At time t, the system real-time load L_t is the sum of the load corresponding to the p class user behaviors, namely $L_t = \sum_{i=1}^{p} L_i$, where L_i represents the system load corresponding to the user behaviors of the group i.

Definition 9 System processing capability L_{HC}. It is the system load corresponding to the user behaviors which can be processed by the system in unit time. If $L_{\max} = L_{HC}$,and $L_t > L_{HC}$, then the system will enter the non-service status.

Definition 10 System load per unit time L_{ut}. It is the system load corresponding to the number of behaviors B_{ut} in unit time. When the system real time load is $L_t \geq L_{safe}$ at t moment, the system load exceeds the processing capacity of the system in unit time. Set $L_{ut} = L_t/t_c$, and $L_{ut} < L_{safe}$, where t_c is the time required to achieve load balancing in the system.

Definition 11 Reconstruction system delay time $\Delta t_d = \sum_{i=1}^{j} t_i$, where t_i is defined as follows:

$$t_1 = \left\lceil \sum_{i=1}^{k_1} L_i / L_{safe} \right\rceil, \quad 1 \leq k_1 < p, \quad \text{when } \sum_{i=1}^{k_1} L_i \leq L_{safe} \text{ and } \sum_{i=1}^{k_1+1} L_i > L_{safe};$$

$$t_2 = \left\lceil \sum_{i=k_1+1}^{k_2} L_i / L_{safe} \right\rceil, \quad k_1 < k_2 \leq p, \text{when } \sum_{i=k_1+1}^{k_2} L_i \leq L_{safe} \text{ and } \sum_{i=k_1+1}^{k_2+1} L_i > L_{safe};$$

$$\cdots\cdots;$$

$$t_j = \left\lceil \sum_{i=k_{j-1}+1}^{k_j} L_i / L_{safe} \right\rceil, \quad 1 < k_j \leq p, \text{when } \sum_{i=k_{j-1}+1}^{k_j} L_i \leq L_{safe} \text{ and } \sum_{i=k_{j-1}+1}^{k_j+1} L_i > L_{safe}.$$

The user interaction behaviors are divided into p classes according to the time sequence characteristics, and L_1,..., L_p are the system load of the p classes. Supposed that the system is in an unstable state, i.e., the system instantaneous load is $L_t > L_{safe}$at t moment. After

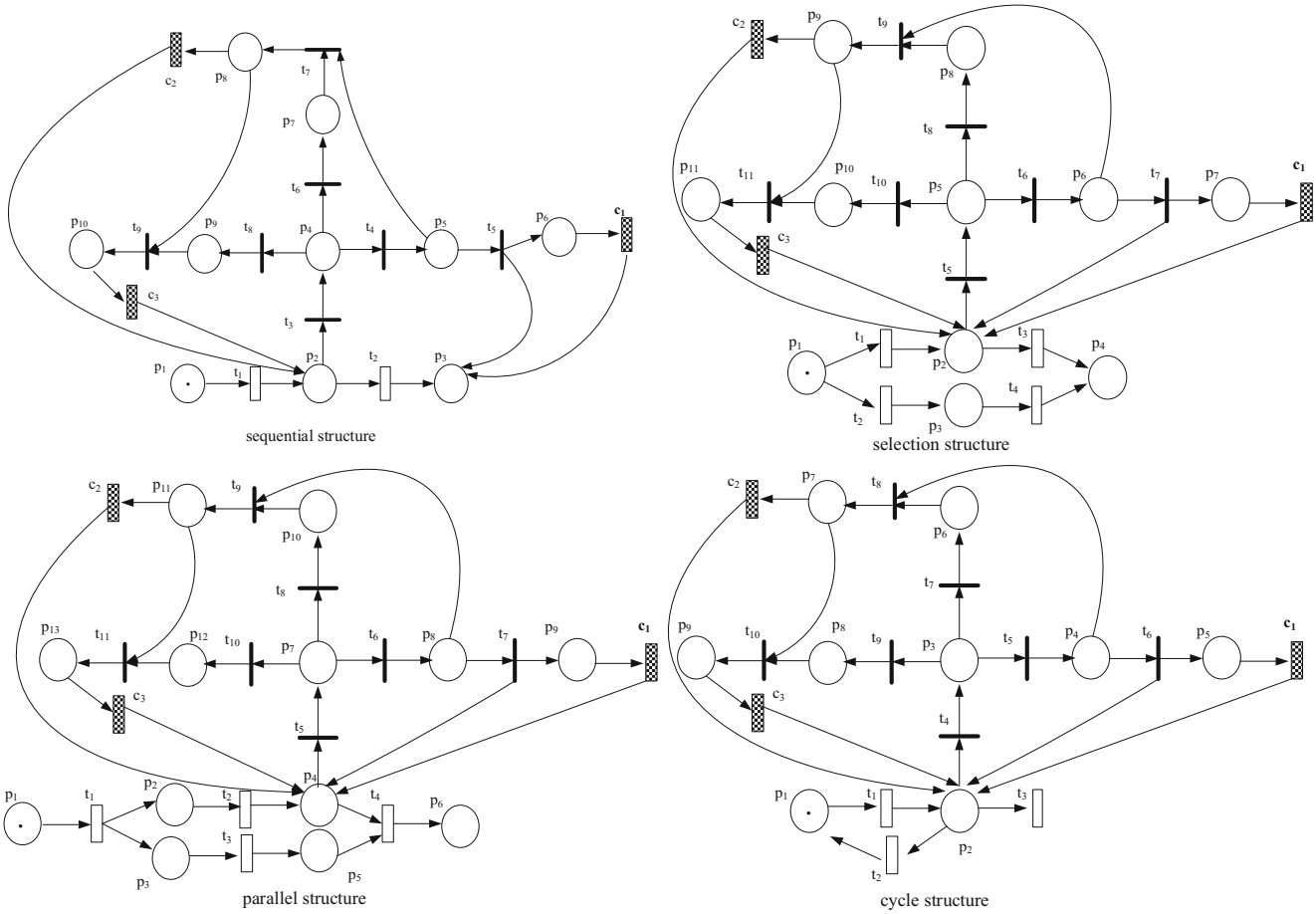


Fig. 1 Four basic structures of the timed stochastic fuzzy Petri Net

reconstruction, the instantaneous system load is $L_{t'} \leq L_{safe}$ at any time t' in the Δt period, and the total system load is equal to L_t in Δt_d time.

Assumption 1 The large-scale network service system itself has a maximum system load L_{max}.

Assumption 2 When the number of user behaviors at a certain time increases sharply, which leads to an abnormal system i.e., $L_t > L_{safe}$, users can be classified according to the user interaction behavior time sequence.

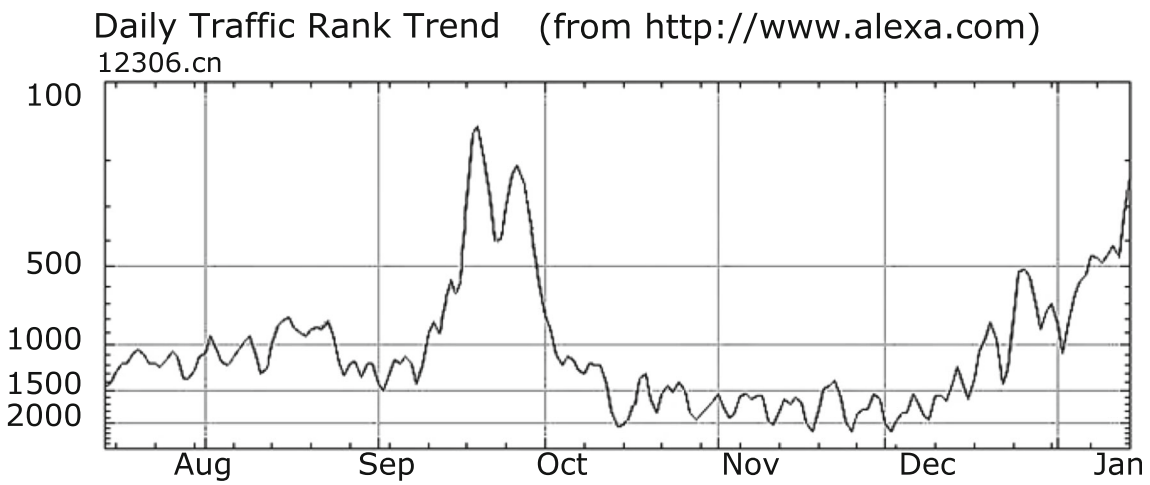


Fig. 2 Train ticket booking website flow chart

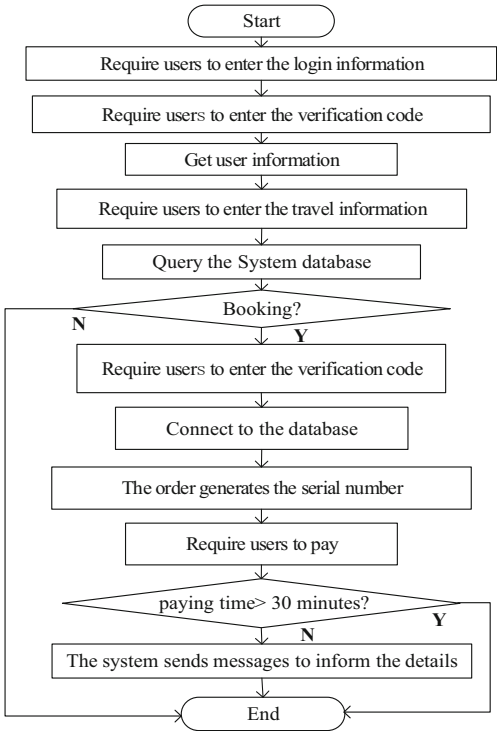


Fig. 3 A booking system flow chart

3.1 Theorem. Load balancing reconstruction

Under Assumption 1 and 2, suppose that the system real-time load is $L_{t_1} > L_{safe}$ at time t_1. The user behaviors are classified accordingly, and the system behavior flows are reconstructed at the system interactions. So the instantaneous load is $L_{t_2} \leq L_{safe}$ at any time t_2 in the delay time Δt_d.

Proof According to def.4, $B_{t_1} \times l > L_{safe}$ is $L_{t_1} > L_{safe}$ at time t_1. The users are divided into p classes, so $\sum_{i=1}^{p} B_{s_i} \times l > L_{safe}$.

Set $\sum_{i=1}^{k} B_{s_i} \times l \leq L_{safe}$ and $\sum_{i=1}^{k+1} B_{s_i} \times l > L_{safe}, 1 \leq k < p$.

If the system dealing with the load $\sum_{i=1}^{k} B_{s_i} \times l$ requires one unit time, then $L_{t_1} - \sum_{i=1}^{k} B_{s_i} \times l = \sum_{j=k+1}^{p} B_{s_j} \times l$, it takes time $\sum_{j=k+1}^{p} \lceil L_j / L_{safe} \rceil$.

So, we obtain $\Delta t_d = 1 + \sum_{j=k+1}^{p} \lceil L_j / L_{safe} \rceil$.

According to def.11, the instantaneous load is $L_{t_2} \leq L_{safe}$ at any time t_2 in the delay time Δt_d. Therefore, the system load L_{t_1} can be balanced at the Δt_d.

4 Petri net model and algorithm for implementing system behavior reconstruction

In this section, the system behavior reconstruction model provides a theoretical support for the adaptive reconstruction process for load balancing in the large-scale network service system. This section will elaborate the implementation of the system behavior reconstruction model in the actual system behavior reconstruction process based on user classification.

4.1 Random fuzzy Petri nets with time delay

Delay Petri nets [20] define the occurrence of changes that needs to be completed by a units of time. This transition issue can be divided into the problem of time transition and immediate transition, Li [21] used the stochastic Petri nets to construct the social network system model. Milinkovic [22] proposed a fuzzy Petri net (FPN) model for estimating train delay. On this

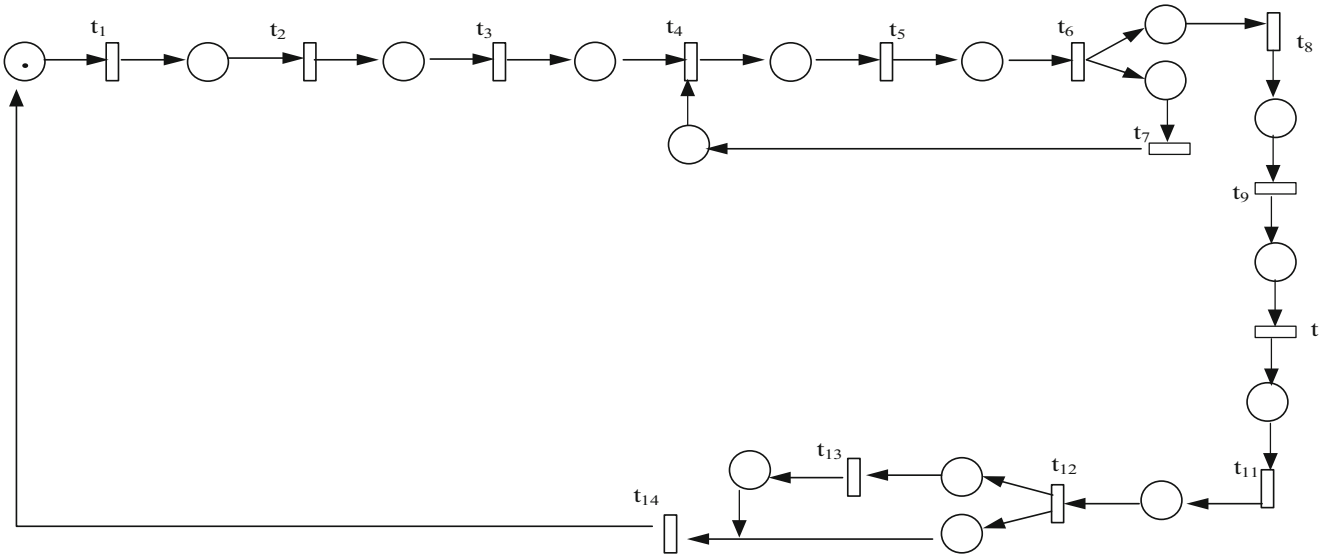


Fig. 4 Petri net model of a booking system

Table 1 Transition description of t_1~t_{14} in Fig. 4

Transition tag	Description
t_1	Require users to enter the login information
t_2	Require users to enter the verification code
t_3	Get user information
t_4	Require users to enter the travel information
t_5	Query the system database
t_6	Whether the user is booking
t_7	Return a query page
t_8	Requires users to enter the verification code
t_9	Connect to the database
t_{10}	The order generates the serial number
t_{11}	Require users to pay
t_{12}	The system judges whether the time is over 30 min
t_{13}	The system sends an e-mail and messages to inform the details
t_{14}	User booking ends

basis, in order to implement the system behavior reconstruction, this paper presents a timed stochastic fuzzy Petri Net.

Definition 12 Random fuzzy Petri net (DSFPN) with delay is a seven-tuple $\sum = (P, T; F, C, DI, \tau, M)$, in which:

(1) P is a set of places, $P = \{p_1,p_2,..,p_n\}(n \geq 0)$, and the number of tokens in a place represents the number of user actions. The number of users arriving at the system over a period of time is subjected a to Poisson distribution;

(2) T is a set of transitions, $T = T_t \cup T_i$, $T_t \cap T_i = \varphi$, where time transition set $T_t = (T_1,T_2,...,T_k)$ includes the transitions of service behaviors; and instantaneous transition set $T_i = (T_{k+1},T_{k+2},...,T_{k+i})$ $(k \geq 0, i \geq 0)$ includes the service transitions which are triggered by the system load beyond the warning point;

(3) C is the control service transition set, $C = \{c_1,c_2,..,c_m\}(m \geq 0)$;

(4) DI is the time function on the transition set, $DI{:}C \rightarrow R_0$. For $t \in C$, $DI(t) = a$, it indicates that the occurrence of the transition t requires a units of time to complete;

(5) F is a directed arc set, where $F = F_T \cup F_C$, $F_T \subseteq (P \times T) \cup (T \times P)$, $F_C \subseteq (P \times C) \times (C \times P)$;

(6) τ is a function on the transition set, which represents the triggering threshold of the transition, and its range is $[0,\infty)$.

4.2 Four basic structures of DSFPN model based on user classification

The system model based on Petri net is composed of four basic structures including sequence, parallel, selection and circulation [23]. Therefore, the following four basic structures of large-scale network service systems are modeled by the

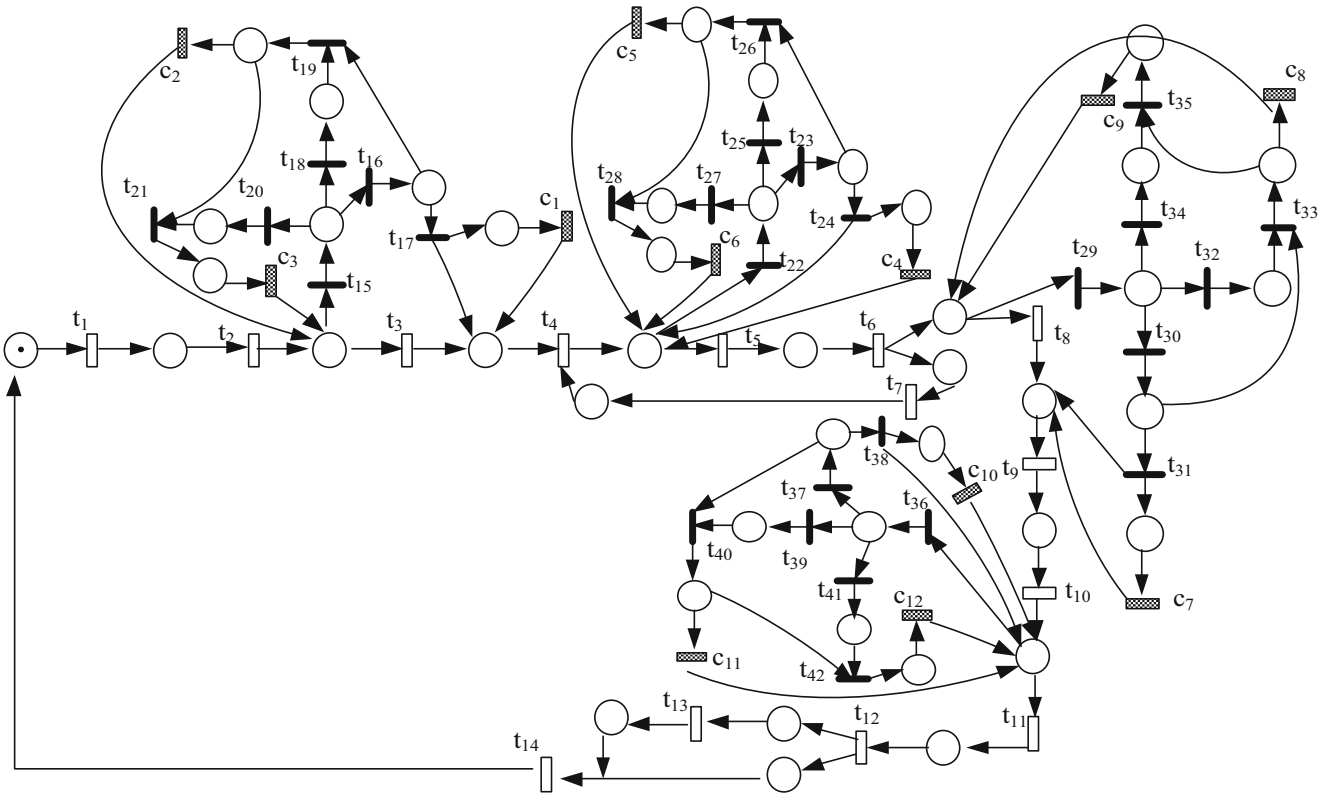


Fig. 5 Random fuzzy Petri net model of the booking system with the delay

Table 2 Transition description of t_{15}~t_{42}, c_1~c_{12} in Fig. 5

Transition tag	Description
t_{15}、t_{22}、t_{29}、t_{36}	Determine whether the load is overloaded
t_{16}、t_{23}、t_{30}、t_{37}	Judge that the speed of user behaviors belongs to the slow group
t_{17}、t_{24}、t_{31}、t_{38}	Judge the load of the slow group
t_{18}、t_{25}、t_{32}、t_{39}	Judge that the speed of user behaviors belongs to the moderate group
t_{19}、t_{26}、t_{33}、t_{40}	Judge the load of the moderate group
t_{20}、t_{27}、t_{34}、t_{41}	Judge that the speed of user behaviors belongs to the extremely fast group
t_{21}、t_{28}、t_{35}、t_{42}	Judge the load of the extremely fast group
c_1~c_{12}	Judge the delay time of three groups

timed stochastic fuzzy Petri net. According to the user membership function, the divided groups of the user behaviors are different. Here, in order to describe the model conveniently, the user behaviors are divided into three groups without the loss of generality, as shown in Fig. 1.

There are three kinds of transitions represented in Fig. 1. The first is the system behavior transition represented by white rectangles. The second is the instantaneous transition for judgment represented by a black line. When the system satisfies the judgment condition, the instantaneous judgment transition is triggered. If the original behavior transition and the judgment transition are in a conflict, the instantaneous transition is triggered in priority. The third is the control transition represented by the shadow rectangle, which controls the delay of each behavior group according to the system load.

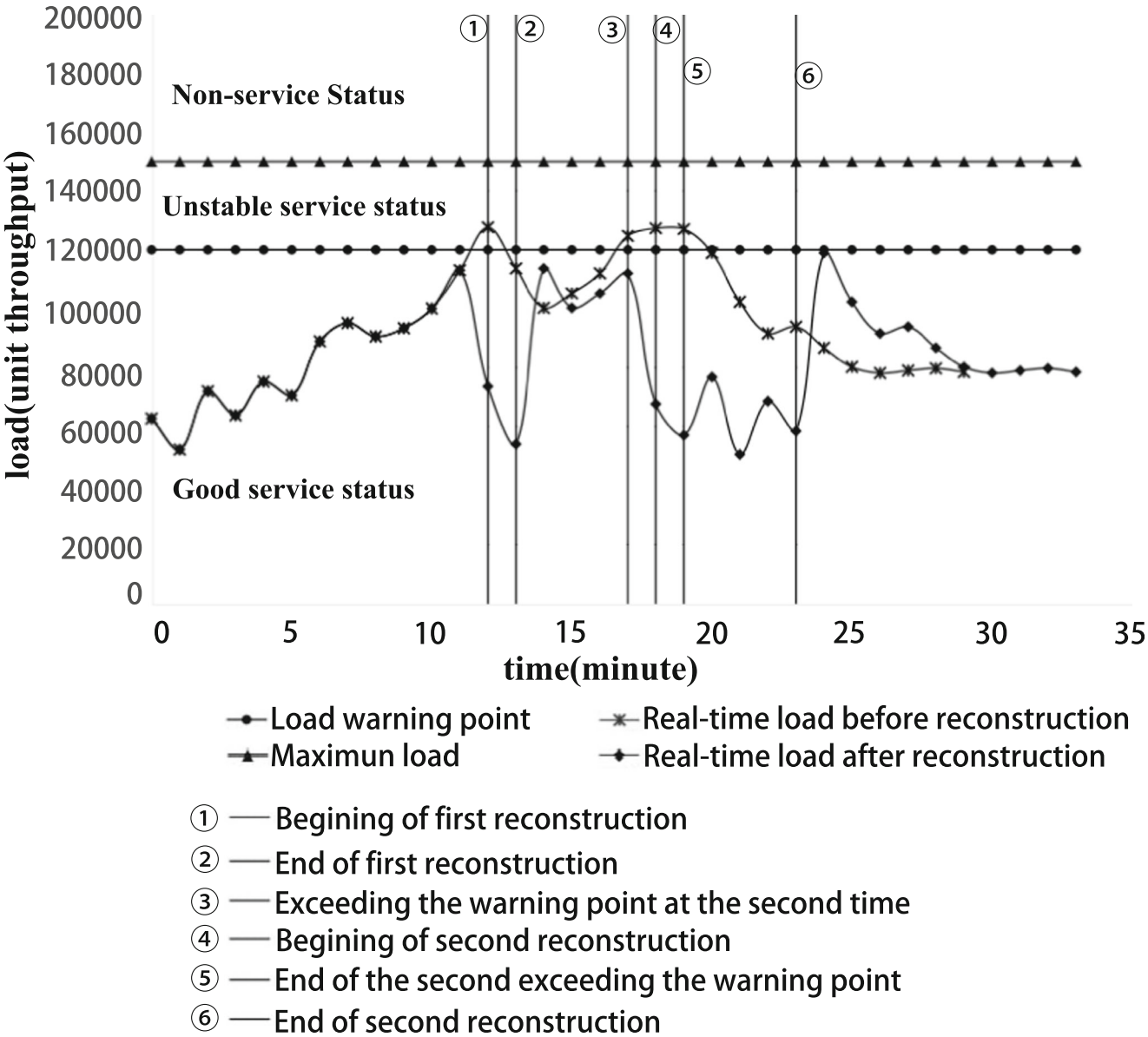


Fig. 6 The first group of experimental results

We take the sequential structure as an example, since the other three cases are nearly similar. Petri net is used to model the behavior of the large-scale network service system. If the key interactive behavior node is in a sequential structure and the system load exceeds the warning point before the node is executed, the system is reconstructed as a DSFPN model to classify the system behaviors.

In Fig. 1, p_1~p_{10} is the place set that represents a state. When the user behavior load submitted to the system exceeds the warning point of the system load, the behavior transition t_2 and immediate transition t_3 face a conflict. Because the priority of the immediate transition t_3 is higher than the behavior transition t_2, the immediate transition t_3 will be triggered. According to the user interaction speed, the user behaviors are divided into three groups as slow, medium and fast speed. Immediate transitions t_4, t_6 and t_8, respectively decide the group to which a behavior belongs. Immediate transitions t_5, t_7, and t_9 respectively judge the relationship between the corresponding load of three behavior groups and the warning point. Control transitions c_1, c_2, c_3 control the delay of three behavior groups.

4.3 DSFPN algorithm

In the above Petri net model, the reconstructed flow will be activated when the tokens in behavior places exceed a certain value, i.e., when the system load exceeds the safe load. According to the definitions 1~11, the DSFPN can obtain the system load, each classified group load and required delay accordingly. The corresponding algorithm of system behavior reconstruction is described as follows.

Algorithm DSFPN:

Input：B_t，p，m，l；*//Bt is the number of user behaviors which detect or work out in time t, p is the number of user groups, m is the number of the user interaction behavior sequence time, l is the load of a user request behavior.*

Output：*the system load in* Δt_d *time*

（1）*while t>0*

（2）$L_t=B_t*l$；*//It is the system load corresponding to the total number of behaviors submitted by the users within time t*

（3）*if* $L_t>L_{safe}$ *then* if the system real-time load exceeds the warning point, and the system enters the unstable service state, then the system begins to perform the reconstruction model;

（4）$i\leftarrow 1$；

（5）*while* $i\le B_t$ *//Calculate the degree of each interactive behavior membership*

（6）$j\leftarrow 1$；

（7）*while* $j\le p$

（8）$\mu_{U_i}(j)=\sqrt{\sum_{k=1}^{m}(u_{ik}-s_j)^2}$；

（9）$j=j+1$；*end while*

（10）$i=i+1$；*end while*

（11）$i\leftarrow 1$；

（12）*while* $i\le B_t$

（13）$d(u_i,s_j)=\min(\mu_{U_i}(j))$；*//Calculate the group which each behavior belongs to*

（14）$i=i+1$；*end while*

（15）Control the behavior classification by the Petri net;

（16）$k\leftarrow 1; c\leftarrow B_{s1}*l; sum\leftarrow 0;$

（17）*while* $sum\le L_{warning}$；*//Calculate the delay time* Δt_d *required for equalizing the load* L_t

（18）$sum=sum+c$；$k=k+1$；*end while*

（19）$\Delta t=1$；*//Calculate the load processed by the system per unit time*

（20）$\Delta t_d\leftarrow 0$；$t'\leftarrow 0$；$SUM\leftarrow 0$；

（21）*while* $k\le p$ *//Calculate the time required for residual load balancing*

（22）*while* $sum< L_{safe}$

（23）$SUM=SUM+L_k; k=k+1$；*end while*

（24）$t'=1$；

（25）*if* $L_k\ge L_{safe}$

（26）$t_k=\lceil L_k/L_{safe}\rceil$；

（27）$t'=t'+t_k$；$k=k+1$；*end while*

（28）$\Delta t_d=\Delta t+t'$；

（29）Print the load condition after the system balances the load within time Δt_d;

（30）$t=t+\Delta T$; *//The detection interval is* ΔT

（31）*end while*

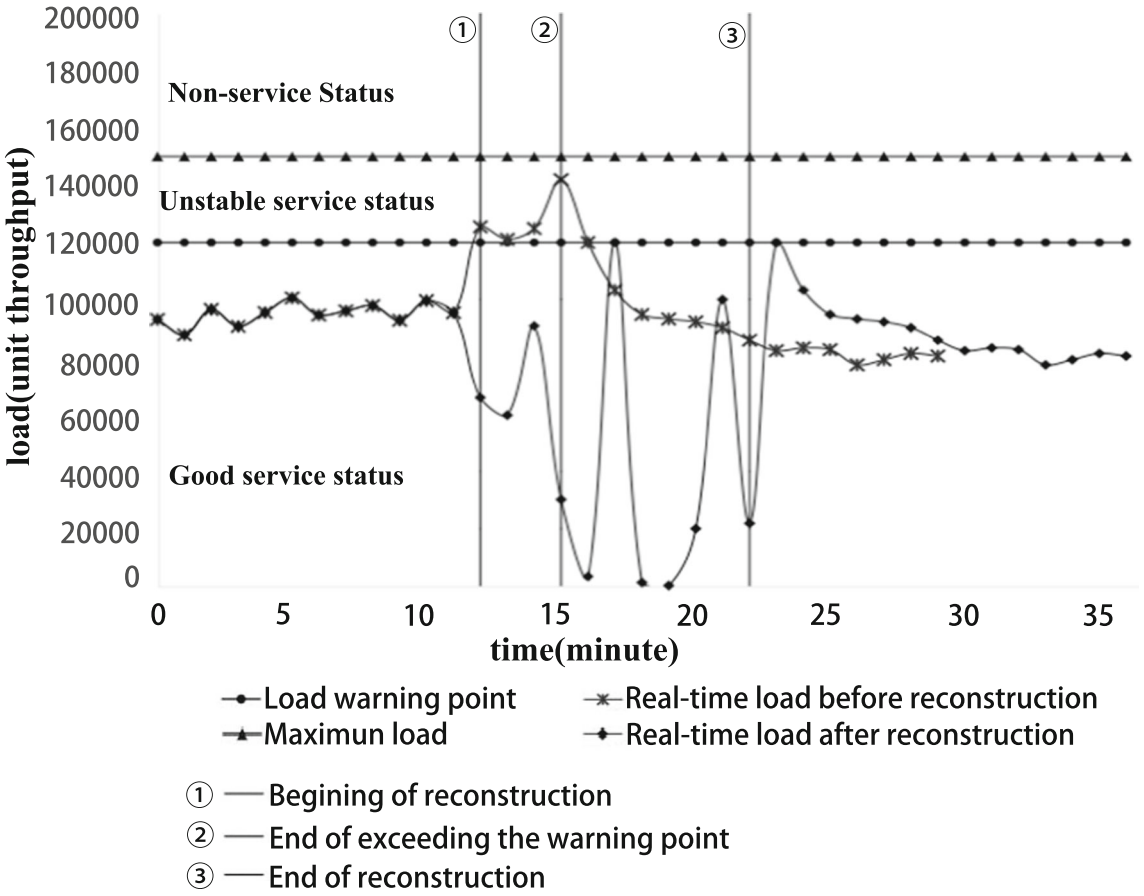


Fig. 7 The second group of experimental results

5 Examples and experiments

5.1 The DSFPN model of a booking system

Online shopping systems and ticket booking systems are typical representatives of the large-scale network service systems, such as Taobao and 12,306 ticket system. Online booking system usually undergo rapid expansion of user groups during seasonal periods such as holidays (Fig. 2). The system will be overwhelmed by this state, this may even paralyze the system. This situation demands necessary modifications in the service process to accommodate the changes occurring in the system load. The process flow in a ticket booking system is simulated (Fig. 3).

Now, an appropriate Petri net model is constructed according to the process flow in the ticket booking system, as shown in Fig. 4. The notations of the behavior transitions in Fig. 4 are shown in Table 1. The sequential user interaction behaviors in this system can be presented as follows: login, query, booking and paying. These interactive behaviors are reconstructed according to the defined DSFPN model, as shown in Fig. 5. The notations of the transitions of t_{15}~t_{42}, c_1~c_{12} in Fig. 5 is shown in Table 2.

5.2 Experimental design

The experiment simulates the system process based on the the booking system flow chart, as shown in Fig. 3. It detects and collects the number of user actions and records the time of the each of the user interaction behaviors. The tokens in the places present the user amount. We use a data generator to continuously increase the number of user actions, which is responsible for increasing the system load. The data for simulation is generated according to the flow chart of the 12,306 train booking system, as shown in Fig. 2, so that the experiment replicate the actual traffic characteristics such as gradually changing user behavior, little changes in the user behavior and suddenly changing user behavior.

According to the traffic characteristics, user behavior is simulated under three different types of load service states including [0,120000) as a good service state, [120000150000) as an unstable service state, [150000,+∞) as an unavailable service state, and it has been assumed that the user behavior is divided into three categories. Now in this simulation environment, we train the system data into the reconstruction algorithm. We implement the simulation system in C++ and use the drawing tool TeeChart to interpret the experimental renderings.

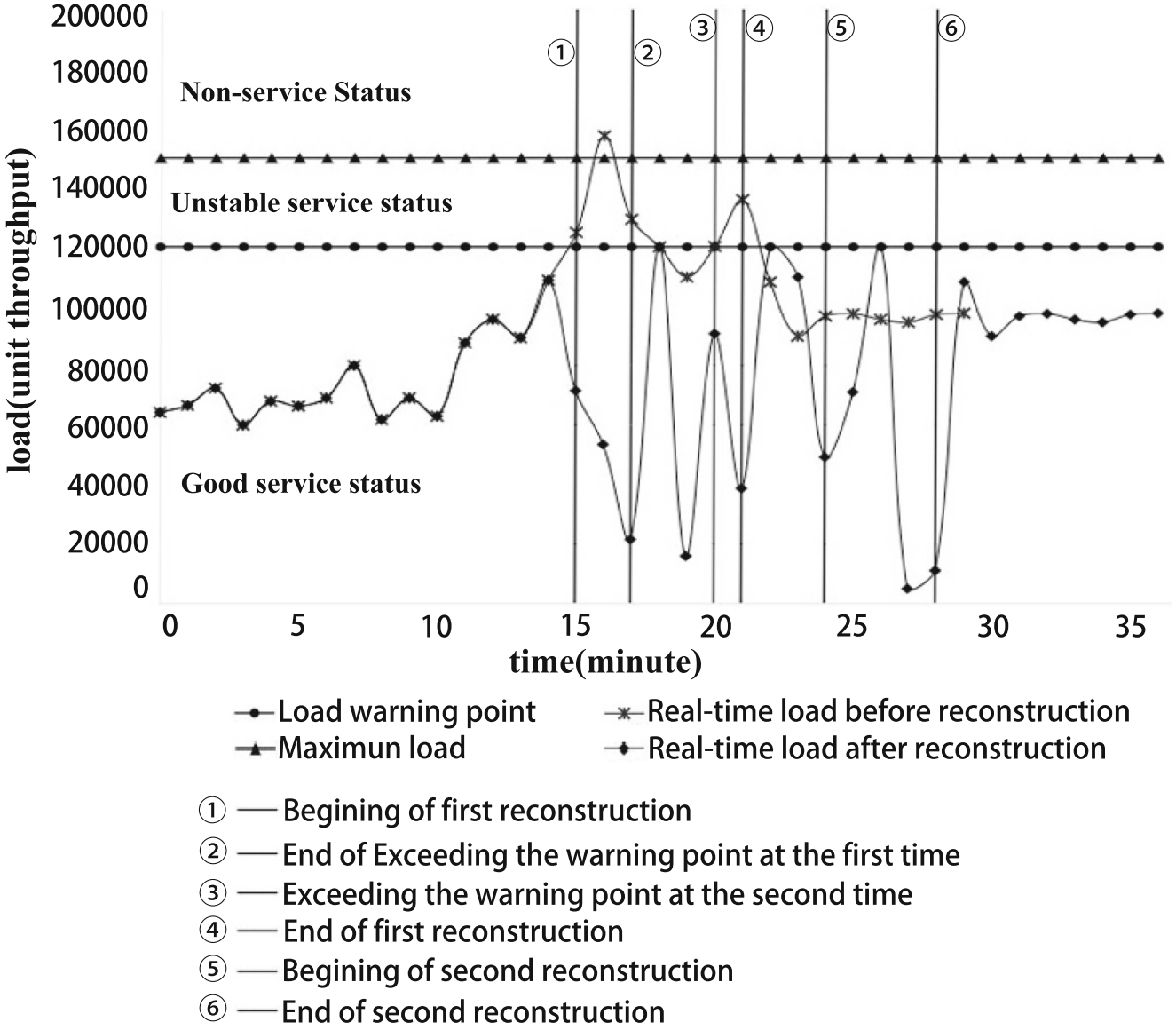


Fig. 8 The third group of experimental results

5.3 Experimental results analysis

The first set of simulated experimental data is applied to the load balancing algorithm of the system behavior reconstruction process; the experimental results are shown in Fig. 6, where the real-time load changes with time are illustrated. The real-time load exceeds the warning point, but it is not obvious, reflecting that the change is not significant. When the real-time load in the system exceeds the warning point, the reconstruction model is triggered and executed. So the user behaviors are divided into three categories and the system load is balanced. The load at any time does not exceed the warning point after the point of equalization, and the system is in good service state.

The second experimental results are shown in Fig. 7. The real-time load exceeds the warning point significantly, but it does not exceed the maximum load which the system cannot withstand. That is, the magnitude of the change is significant. When the real-time load exceeds the warning point, the reconstruction model is triggered and the system load is balanced.

The third experimental results are shown in Fig. 8. The real-time load exceeds the maximum load, that is, the magnitude of the change is huge. When the system real-time load exceeds the warning point, the reconstruction model is executed and the system is in good service state at any time.

From the above three groups of experimental results, if the system real-time load exceeds the warning point, the system enters into the unstable service state, and the system reconstruction model is triggered. At this time, the user behaviors are classified by the time features of the behavior sequence, and the system load is balanced by controlling the interaction time

of each type of user group. Therefore, the system load does not exceed the warning point at any time. The experimental results show that the system behavior reconstruction model based on time features of the user behavior sequence can effectively balance the system in good service condition at any time.

6 Conclusions

This paper proposes a system behavior reconstruction model based on the user interaction time sequence characteristics, with the aim of resolving the system overloading issue resulting from the rapid growth of user behaviors large-scale network service systems, by the way of delaying user behavioral time. Furthermore, a reconstruction algorithm has been developed based on random fuzzy Petri net with imposed delay.

The user behaviors have been classified based on a membership function and a membership criteria, which provided the basis for constructing the system behavior reconstruction model. In the actual service systems, the behavioral flow of different user groups has been constructed by the system behavior reconstruction model and an algorithm based on randomized fuzzy Petri net with delay has been implemented. The proposed model for the balancing the system load guarantee that the system is always in good running state. As a future work, we plan study the potentials of adaptive refactoring system in effectively balancing the system load.

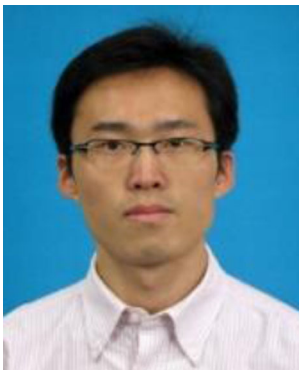
Acknowledgements This work was supported by National Natural Science Foundation of China (No. 61472004, 61602109), Shanghai Science and Technology Innovation Action Plan Project (No.16511100903), and by The Key Laboratory of Embedded System and Service Computing of Tongji University of Ministry Education (2015).

References

1. Zhaolei D, Zhimin G (2008) Dynamic load balancing in web cache cluster. 7th International Conference on Grid and Cooperative Computing, pp 147–150
2. Yang L, Weizhe J, Youbo L (2017) A sliding window-based dynamic load balancing for heterogeneous Hadoop clusters. Concurr Comp Pract E 29(3). https://doi.org/10.1002/cpe.3763
3. Saxena S, Khan MZ, Singh R (2012) Performance analysis in distributed system of dynamic load balancing using fuzzy logic. Spring Congress on Engineering and Technology, IEEE, pp 6–12
4. Zouina D, BenMussa SA (2017) Load balancing aware SDMA-based beaconing approach in vehicular ad hoc networks. Ann Telecommun 72(3–4):189–197
5. Hwang ST, Jung NS (2002) Dynamic scheduling of web server cluster. Proceedings of the 9th International Conference on Parallel and Distributed System, pp 563–568
6. Liu F, Chen Y, Wong WS (2016) An asynchrous load balancing scheme for multi-server systems. //2016 I.E. 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference: IEEE Press, pp 20–22
7. Yong W, Xiaoling T (2016) A dynamic load balancing method of cloud-center based on SDN. China Commun 13(2):130–137
8. Kallel S, Rodruigez IB, Drira K (2016) Adaptive and reconfigurable software systems and architectures. J Syst Softw 122:342–343
9. Zhang J, Khalgui M, Li Z et al (2015) Reconfigurable coordination of distributed discrete event control systems. IEEE Trans Control Syst Technol 23(1):323–330
10. Wattebled P, Diguet J-P, Dekeyser J-L (2012) Membrane-based design and management methodology for paralleldy namically reconfigurable embedded systems. //7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip: IEEE Press, pp 1–8
11. Santos R, Venkataraman S, Kumar A (2015) Dynamically Adaptive Scrubbing Mechanism for Improved Reliability in Reconfigurable Embedded Systems. //2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp 1–6
12. Yang F, Shen L-X (2015) Reconfigurable architecture model based on layered hypergraph. //Chinese Automation Congress:IEEE Press, pp 27–29
13. Khalgui M, Mosbahi O, Li ZW, Hanisch H-M (2011) Reconfigurable multiagent embedded control. IEEE Trans Comput 60(4):538–551
14. Wang G, Zhang X, Tang S, et al (2016) Unsupervised clickstream clustering for user behavior analysis. //SIGCHI Conference on Human Factors in Computing Systems
15. Xu L, Mladen K (2008) Implementing fuzzy reasoning Petri-nets for fault section estimation. IEEE Trans Power Delivery 32(2):676–685
16. Zoran K, Pece M (2014) Hybrid fluid modeling approach for performance analysis of P2P live video streaming systems. Peer Peer Netw Appl 7(4):410–426
17. Lu MT, Tzeng GH, Cheng H et al (2015) Exploring mobile banking services for user behavior in intention adoption: Using new hybrid MADM model. Serv Bus 9(3):541–565
18. Richardson M (2008) Learning about the World through Long-Term Query Logs. ACM Trans Web 2(4):1–27
19. Iglesias JA, Angelov P, Ledezma A, Sanchis A (2012) Creating evolving user behavior profiles automatically. IEEE Trans Knowl Data Eng 24(5):854–867
20. Ahangarani FA, Abbas D (2016) Continuous-Time Delay-Petri Nets as A New Tool to Design State Space Controller. Inf Technol Control 5(4):401–U66
21. Linyu L, Wu Z, Zhiguo H, Long Z (2016) Stochastic Petri Net-based performance evaluation of hybrid traffic for social networks system. Neurocomputing 204:3–7
22. Sanjin M, Milan M, Slavko V, Milos I, Norbert P (2012) A fuzzy Petri net model to estimate train delays. Simul Model Pract Theory 33:144–157
23. Chuang L, Yang Q, Fengyuan R (2002) Performance equivalent analysis of workflow systems based on stochastic Petri net models. Engineering and Deployment of Cooperative Information Systems, First International Conference, EDCIS 2002, Beijing, China

Zhaohui Zhang received the B.S. degree in Computer Science from Anhui Normal University, Wuhu, China, in 1994, and became a teacher at the University. He completed his master's courses program of University of Science and Technology of China from 1999 to 2000. Respectively, he received the Ph.D. degree in Computer Science from Tongji University, Shanghai, China, in 2007. He was a professor with Anhui Normal University before 07/2015. Currently, he is a Professor with the School of Computer Science and Technology, Donghua University, Shanghai, China. His research interests include network information services, service computing and cloud computing.

Lina Ge was born in 1993. She received the B.S. degree Computer Science from Huaibei Normal University, Huaibei, China, in 2014. She is a M.S. candidate of Anhui Normal University. Her research area includes clouding computing and service computing.



Pengwei Wang received the B.S. and M.S. degrees in Computer Science from Shandong University of Science and Technology, Qingdao, China, in 2005 and 2008, respectively, and the Ph.D. degree in Computer Science from Tongji University in 2013. He finished his postdoctoral research work at the Department of Computer Science, University of Pisa, Italy, in 2015. Currently, he is an Associate Professor with the School of Computer Science and Technology, Donghua University. His research interests include service computing, cloud computing, and Petri nets.

Xinxin Zhou was born in 1993. She is a M.S. candidate of Donghua University. Her research area includes clouding computing and service computing.