Research Journal of Applied Sciences, Engineering and Technology 5(24): 5561-5565, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: September 30, 2012 Accepted: December 12, 2012 Published: May 30, 2013

Research on Incomplete Transaction Footprints in Networked Software

[1]Junfeng Man, [2]Cheng Peng, [3]Qianqian Li and [1]Changyun Li
[1]College of Computer and Communication, Hunan University of Technology, Zhuzhou 412007,
[2]School of Information Science and Engineering, Central South University, Changsha 410083,
[3]Department of Information and Engineer, Hunan Chemical Vocation Technology College,
Zhuzhou 412004, China

Abstract: In networked software, interactive behaviors of software entities generate lots of behavioral footprints, some of them may lose tokens or tokens are useless. The paper studies the constructing process of State Transition Model (STM) in which the process of incomplete transactions are satisfied with Markov property, it is pointed that the STM are originally extracted from behavior log files generated by the runtime behaviors of networked software. The contributions of this paper are to mark the partly tokenized behavior footprints in the STM through Maximum Flow (MF) algorithm, then find the original source for each behavior footprint. The experiment results indicate that the maximum flow algorithm can accurately turn the partly tokenized behavior into complete footprint sequences.

Keywords: Behavior footprints, incomplete transaction maximum flow algorithm, networked software, state transition model

INTRODUCTION

The emergence of the Internet has made software from the static, closed and controllable environment into open, dynamic and uncontrollable condition, in order to adapt to this trend, traditional centralized software system gradually turns into the distributed networked software (Yang *et al*., 2002); the dynamic changes of self state and interactive environment of the runtime networked software is called dynamic evolution. Software dynamic evolution is the process of online adjustment in order to reach the hope state, which happens in runtime, consists of a series of complex activities, e.g., dynamic updating, adding or deleting software components, system structure dynamic configuration etc (Kramer and Magee, 1990).

However, we only consider the changes of software entities structure, seldom notice the running state of software entities as individuals in interactive activities and the interaction behaviors they generate. At present, the software failure and fault are becoming more and more serious, not only increase the cost of use and maintenance, but also make people lost the confidence for software (Chen *et al*., 2003). The recent studies show that fault and failure of software are largely due to the software behaviors can not meet the user' expectation.

We can get the original message of software behavior (Li *et al*., 2009) through monitoring, but current monitoring technology cannot enhance the software reliability at once, still need further effectively analysis of the software interaction. Generally, we can use the correlator which is the open-group ARM instrumentation or token to track transaction circulation. But, many transactions are often executed in parallel in networked environment, such events generated by these transactions mix together, which results in the tokens of transaction behavior footprints lost or can not be used, thus software system may not accurately and correctly identify the only source of every footprint. These partly tokenized behavior footprints cause great obstacles for subsequent behavior analysis, behavior prediction and other operation. Therefore, tokenizing the interaction entity behavior is in the important position for guaranteeing the reliability of networked software.

STATE TRANSACTION MODEL

The concept and definition: Let $f_x(x)$ be the Probability Density Function (PDF) of a continuous random variable X and $\bar{F}_X(x) := P[X > x]$ its complementary cumulative distribution function (CCDF).

For an undirected graph G(V, E), if $V = X \cup Y$ and $X \cap Y = \Phi$, such that every edge *E* connects a vertex in *X* to one in *Y*, *G* is bigraph, denoted as G = (X, Y, E).

For a directed acyclic graph G = (V, E, C), *V* denotes vertex set, *E* denotes directed edge set and

Corresponding Author: Junfeng Man, College of Computer and Communication, Hunan University of Technology, Zhuzhou 412007, China

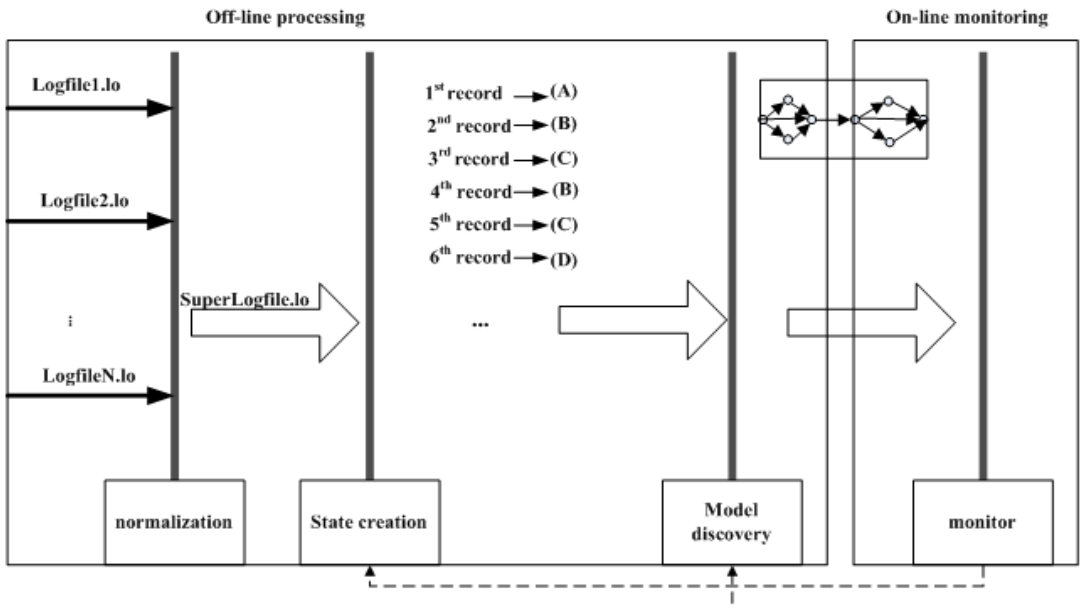

Fig. 1: Discover state transition model

denotes capacity, then G is called network flow graph (Zhang *et al*., 2003). If C_{ij} denotes the allowed maximum flow from V_i to V_j, x_{ij} denotes the real flow at this arc, there is $0 \le x_{ij} \le C_{ij}$. In the network, except for the start and the end vertices, any other vertices, the total of inflow is equal to the total of the outflow, namely, $\sum x_{ij} = \sum x_{ji}, i \ne s,t$. At this time, $f = x_{ij}$ is known as feasible flow. The maximum network flow can be described as:

$$\max f s.t. \begin{cases} \sum_j x_{ji} - \sum_i x_{ij} = \begin{cases} -f, i = s \\ 0, i \ne s,t \\ f, i = t \end{cases} \\ 0 \le x_{ij} \le C_{ij} \end{cases} \quad (1)$$

A Markov chain is a sequence of random variables X_1, X_2, X_3, ... with the Markov property, namely that, given the present state, the future and past states are independent (Lee and Ho, 2006).

Let S_i , I = 0 , … , N_s denote the ith state of the process and let $T_{i,j}$ denote the (random) time to transition from state S_i to state S_j . A process is said to be semi-Markov if the sequence of states visited is a Markov chain, with transition probability matrix P = [P(i, j)] and each transition time $T_{i,j}$ is a random variable that depends only on the states S_i and S_j involved in the transition.

Discover state transition model: Figure 1 depicts the process of discovering state transition model from the history log files. Our approach consists of two phases: off-line processing and online monitoring, the latter refers to on online monitoring of transaction instances.

The monitoring information is inputs for:

- Online system log files, where footprints left by ongoing transaction instances are being recorded
- The transaction model discovered through offline processing. Using these inputs, the monitoring system generates dynamic execution profiles of ongoing transaction instances that allow their status to be tracked at individual and aggregate levels.

State creation: Analyzing the records in the log files and creating candidate states will typically involve some form of clustering. For each new log record to be processed, the record is tokenized and placed in a "bucket" of records with the same number of words; we refer to this number as the dimensionality n of the record. This process creates a partitioning of the sample space of log records into sample subspaces Ω_n n = 1, …, N, of log records. As shown in Fig. 2.

Model discovery: State transition follows Markov, that's to say, the transition probability of each transaction instance under each state, only depends on the current state and the next state. As shown in Fig. 3, except the behavior footprints at state A has tokens (express by different pictures), others behavior footprints at the intermediate state only partly tokenized, the aim of the paper is to find the original source for each behavior footprint that lost token.

The transition model is a directed acyclic graph (DAG), which consists of a start states set and an end states set. At start state, the transaction goes into the model; at the end state, the transaction quits the model. The transition time between each state is Independent

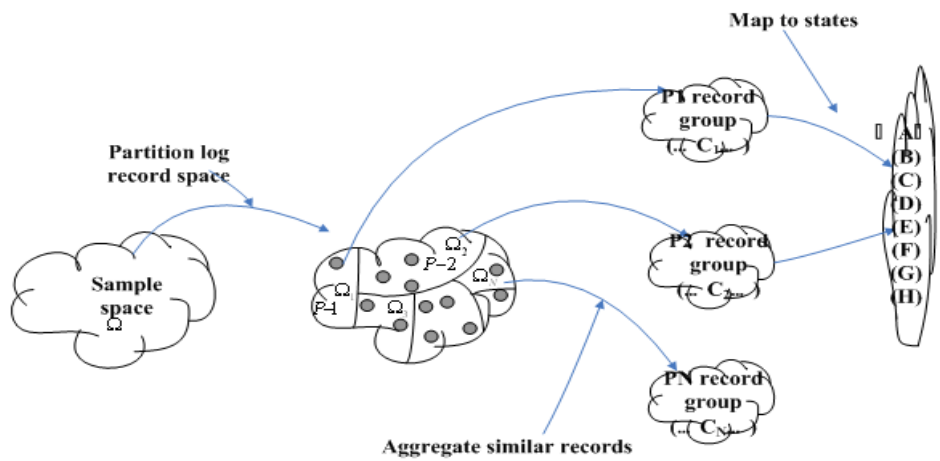

Fig. 2: State creation

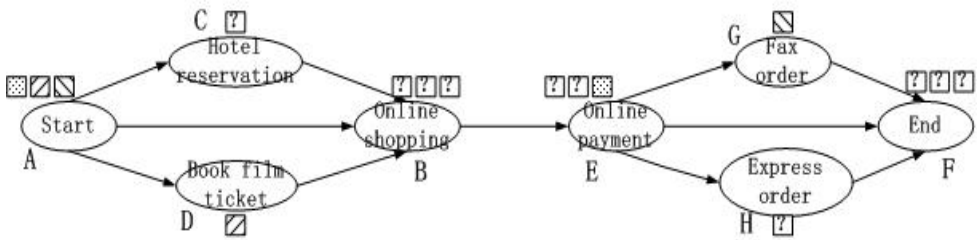

Fig. 3: The instance of state transition model

and Identically Distributed (I.I.D.). Each state S_k in the model, any valid matching can be denoted by the set of permutation vector π_k. Set Y_k denote footprint timestamp vector of state S_k. When the monitoring engine received behavior footprints by the correct time order, Y_k ordered by the ascending, so the latest events arranged at the rear. According to permutation vector π_k, let $Y^{\pi k}{}_k$ denote the permutation of Y_k. For convenience, at start state, we assign tokens to footprints Y_0.

When the Probability Density Function (PDF) of transaction transition time f_T is known, we could transfer the problem of quantitative tracking into figure out the probability of all instances properly match their footprints by MLR. So, for N_{S+1} states, transaction ML tracing simplified as finding the N_s sets of permutation vector:

$$[\hat{\pi}_1^{ML}, \ldots, \pi_{N_s}^{ML}] := \arg\max_{\pi_1,\ldots,\pi_{N_s}} P(Y_1^{\pi_1}, \ldots, Y_{N_s}^{\pi_{N_s}} \mid Y_0) \quad (2)$$

As the transition time of each transaction is Semi-Markov Process (SMP), the transition time of different transactions is also independent, that is to say, the transition time between states only depends on the current state and the next state, according to this, we can splice the multi-state system into two parts: the first one is the set of predecessors and the other one is the set of successors, So the multi-state system model described in the formula (1) can be transformed in:

$$P\left(Y_1^{\pi_1},\ldots,Y_{N_S}^{\pi_{N_S}} \mid Y_0\right) = \prod_{m>0} P\left(\bigcup_{S_k \in B_m} Y_k^{\pi_k} \mid \bigcup_{S_l \in P(B_m)} Y_l^{\pi_l}\right) \quad (3)$$

Here define a partition of states (B_1 , … , B_m) , $m \geq 0$ and $B_0 = S_0$. States in any two sets in the partition do not share a common immediate predecessor, it is $p(S_k) \cap p(S_l) = \Phi$, $\forall S_k \in B_m, S_l \in B_j, m, j \neq 0$. So, bipartite graph can be described as ($p(B_m)$, B_m), where start state is $p(B_m)$, end state is B_m. According to the definition of state partition, all bipartite graphs are disjoint, a state cannot occur in two systems.

IDENTIFY PARTLY TOKENIZED TRANSACTION

It's easy to prove that the maximal matching of bigraph $G = (X, Y, E)$. is corresponding to the maximal network flow of graph $\acute{G} = (s, t, X, Y, E, C)$., C is capacity set, as is shown in Fig. 4. When the network flow achieve maximum, if the capacity at (x_i, y_j) is one, which means x_i and y_j is the perfect matching. We

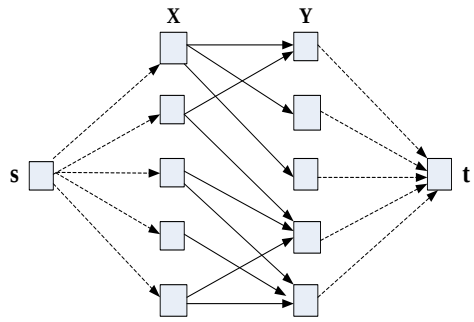

Fig. 4: Transfer bipartite into network flow diagram

adopt the algorithm of Ford-Fulkerson to solve the problem of bigraph maximal matching, this algorithm is introduced in literature 9.

Note the start time and end time in the two state sets of bigraph subsystem as t_{out} and t_{in} separately and expressed by node v_{out} and v_{in}, we get node set V_{out} = $\{v_{outi} / i = 1, n\}$ when departing state and node set V_{in} = $\{v_{ini} / i = 1, n\}$ when entering state, n is the total number of all behavior footprints in a single state set of bigraph subsystem. For any node v_{outi} and $v_{in\ j}$, if satisfied with the transition time limit between two states, which means footprint i and footprint j form the matching, link the node v_{outi} and $v_{in,j}$ with directed arc, we get connection arc set:

$$E_c = \{e_{ij} \mid e_{ij} = (v_{outi}, v_{inj}), t_{\min} \le t_{inj} - t_{outi} \le t_{\max}, \forall i, j, i \ne j\} \quad (4)$$

Combining with the three bipartite subsystems and Table 1 mentioned above, we construct three network flow diagrams. Constructing arc sets E_c according to formula (4), the state transition time lower limit is t_{min} = 5sec, upper limit is t_{max}, we calculate the state transition time obey index distribution, uniform distribution and normal distribution situation, finally, we get a network flow diagram corresponding to the bipartite matching.

Table 1: Footprints and corresponding timestamp

	Footprints							
Time stamp	TP01	TP02	TP03	TP04	TP05	TP06	TP07	TP08
Start time	204018	204015	204010	204032	204035	204030	204208	204210
End time	204030	204023	204020	204140	204130	204120	204220	204221
	TP09	TP10	TP11	TP12	TP13	TP14	TP015	TP16
Start time	204230	204233	204218	204410	204400	204603	204340	204610
End time	204340	204322	204350	204523	204510	204710	204520	204715

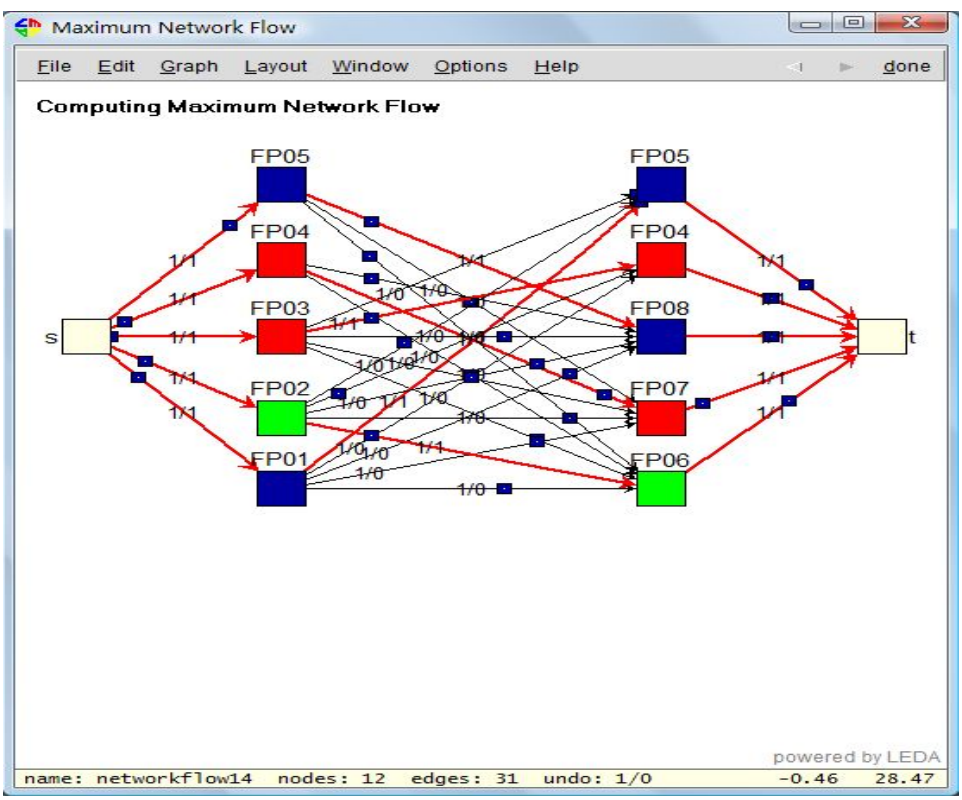

Fig. 5: Maximum network flow matching

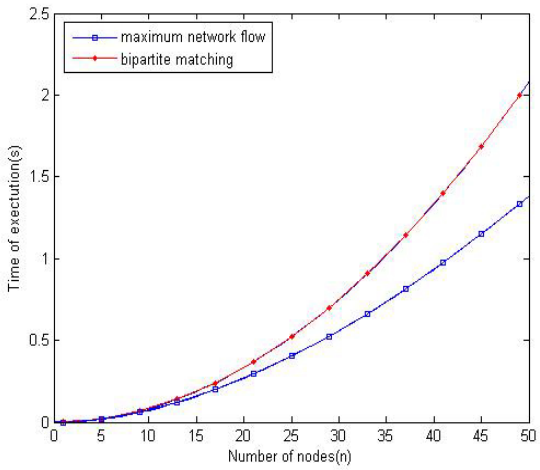

Fig. 6: Algorithm performance comparison

We adopt the network flow method to deal with the matching of the bipartite subsystems in the state transition model; finally, splice the matching results of the three subsystems and form three complete behavior footprint sequences, FP01-FP05-FP08-FP10-FP15, FP02-FP06- FP11-FP12-FP14 and FP03-FP04-FP07-FP09-FP13-FP16. In this way, we find the original source of the footprints that are not tokenized. The simulation experiment results are showed in Fig. 5.

The best time complexity is $O(n(m+n \log n))$, which is strongly polynomial algorithm, n and m denote nodes and edges separately. By adopting maximum weight matching algorithm, no matter the time complexity and space complexity can get the good results. In this paper, we use the algorithm of maximum network flow, the time complexity is $O(nm)$, the two algorithms performance comparison is showed in Fig. 6. Under the same experiment environment, with the nodes in the state transition model increasing, the algorithm we proposed is better than the traditional bipartite maximum weight matching algorithm in dealing with the problem of tokenizing partly tokenized behavior footprints.

CONCLUSION

This study researches on the problem of extracting state transition model from E-commerce trade platform instance under networked environment, combined with monitored log files from the interaction, then remark the partly tokenized footprints generated by each state. Using the state splitting algorithm to divide the state transition model into several bigraph subsystems and transfer each subsystem into network flow graph, matching every footprint under each state, finally, forming the complete footprint sequences. The results of simulation analysis show that our method is feasible.

ACKNOWLEDGMENT

This study is supported by the National Natural Science Foundation of China under grant No. 61171192 and 61170102, the Natural Science Foundation of Hunan province in China under grant No. 11JJ4050 and 11JJ3070, the Education Department Foundation of Hunan Province under the grant No. 11B039, 11W002 and 11C0400.

REFERENCES

Chen, H.W., J. Wang and W. Dong, 2003. High confidence software engineering technologies [J]. Acta Electron. Sinica, 31(12): 1933-1938.

Kramer, J. and J. Magee, 1990. The evolving philosophers problem: Dynamic change management [J]. IEEE T. Software Eng., 16(11): 1-33.

Lee, C.K.M. and G.T.S. Ho, 2006. A dynamic information schema for supporting product life cycle management [J]. Expert Syst. Appl., 31(1): 30-40.

Li, R.J., Z.X. Zhang, H.Y. Jiang and H.M. Wang, 2009. Research and implementation of trusted software constitution based on monitoring [J]. Appl. Res. Comp., 26(21): 4585-4588.

Yang, F.Q., H. Mei, J. Lv and Z. Jin, 2002. Some discussion on the development of software technology [J]. Acta Electron. Sinica, 30(z1): 1901-1906.

Zhang, X.C., G.L. Chen and Y.Y. Wan, 2003. Research on the maximum network flow problem [J]. J. Comput. Res. Dev., 40(9): 1281-1291.