Model Checking and Verification of the Internet Payment System with SPIN

Wei Zhang
Shandong Provincial Key Laboratory of Computer Network
Shandong Computer Science Center, Jinan, China
Email:wzhang@keylab.net

Wen-ke Ma, Hui-ling Shi, and Fu-qiang Zhu
Shandong Computer Science Center, Jinan, China
Email:{mawk, shihl, zhufq}@keylab.net

***Abstract*—With the development of the information technology, using Internet to pay for goods and services has become a very popular application. The traditional forms of payment can not be applied to e-commerce environment. Because of the complexity of online transactions related to capital flows and goods flows, the transaction process requires higher security and reliability. One mature approach for ensuring reliability is to use formal methodologies. In this paper, we employ model checking method to verify the security and reliability of the Internet Payment Systems. A PROMELA model for the System is present. As an important part of our modeling methodology, we translate the Internet Payment System into a simpler model that nevertheless preserves all the essential behavior to be verified. We also propose initial results on the actual verification of the Internet Payment System using SPIN. The result of our work is a complete procedure for the modeling and verification of the Internet Payment System.**

***Index Terms*—the Internet Payment Systems, verification, model checking, SPIN, PROMELA, linear temporal logic**

I. INTRODUCTION

In the traditional business activities, the payment process is mainly classified in paper forms, such as bill payments and cash payments. Recently the Internet has become an essential tool for commerce and financial services. With the help of new communication and information technologies, these services have experienced tremendous growth. The traditional forms of payment can not applied to the e-commerce environment. The reasons are as following: the traditional payment can not be binding and monitor between the participants of the transaction. Quality of the goods, transaction integrity, and requirements of return and replacement can not be reliable guarantee. The Financial Institutions focuses these days to move all payment forms (i.e. transfers, deals, purchases, and bill payments) to electronic form instead of paper form.

Manuscript received July 30, 2011; revised September 5, 2011; accepted December 19, 2011

This work is supported in part by the National Science Foundation of China (Grant No. 61070039).

It is convenient for people to use Internet Payment in online transactions. Because the online transactions are related to both capital flows and goods flows, higher security and reliability is required for the transaction process. In electronic payments, participants may use communication protocols for which there are no transactional variants (e.g. HTTP) and the programs may be deployed in very heterogeneous application environments. For these reasons, electronic payment systems cannot rely on traditional transaction mechanisms [1]. The research on internet payment agreements has been the focus of financial payment system in recent years, in particular, how to ensure the safety and reliability of the system. Software testing is a method to verify the security and reliability of systems. The testing of Internet Payment System has been conducting and researching in recent years. But there are not many related works to verify the logic and design of business processes during the online payment.

One mature approach for ensuring reliability is to use methodologies based on formal methods. In general, this approach consists of constructing a computer tractable description (formal model) of the system design and then using a specific automatic (or semi-automatic) analysis technique to prove or to check the satisfaction of a given set of critical properties [2]. Probably, the most promising formal methods to ensure a prior reliability is model checking [3, 4], Analyzing a payment system with model checking consists of the following steps: (a) construct model of the payment system with the main features that could produce execution errors; (b) specify the reliability properties with a property-oriented language; and (c) produce the reachability graph including all the execution paths for the model in order to check whether these paths satisfy the properties. This technique has been integrated in many academic and industrial oriented tools [5].

This paper discusses how to employ SPIN [5, 6, 7], one of the most powerful and well-known model checking tools, in order to specify and analyze the correctness of protocols for Internet Payment Systems. The rest of the paper is organized as follows. Section 2 provides background material on the Internet Payment

doi:10.4304/jsw.7.9.1941-1949

Systems and the model checking technology. Section 3 presents the modeling process. In Section 4, we give some description of the model, and present some properties of the system. In Section 5 we show the experiments of the verification, and discuss the results. We summarize and discuss related work in Section 6. And in Section 7, we give some conclusions of the works we have done, and point out some further works we would do.

II. BACKGROUND

As theoretic background, we introduce the concepts of Internet Payment Systems, including the classification of the payment systems, general process of the online payment transactions, and some frameworks of the Internet Payment Systems. Also we introduce the theory of model checking technology, SPIN which is one of the model checking tools, and Linear Temporal Logic which is used for applications in software verification.

A Internet Payment Systems

The growing importance of e-commerce and the ever-increasing number of business transaction models has resulted in a plethora of payment systems. Online payments involve communication with a trusted third party (TTP) during payment and in general they are considered as more secure than offline payments that involve only the payer and the payee.

The vast majority of Internet Payment Systems are online systems that perform either:

- Credit-card payments (First Virtual, CyberCash, iKP, Anonymous Credit-Cards).
- Micropayments (NetBill, Millicent, l-iKP, MiniPay and NetCash).
- Or they are used as payment switches (OpenMarket) [1].

In this paper, we focus on the Micropayments, which are widely used in the Internet Payment Systems. The internet payment generally involves three participants: consumers (C), merchant (M) and the Trusted Third Party (TTP). The general process is as follows:

(1)Customers buy goods on the e-commerce sites, and finally decide to purchase; the customers and merchant make the intention of the deal online;

(2)Customers choose to use TTP as a intermediary for the transaction, customers will be paid with a credit card account designated to TTP;

(3)TTP platform will notify the merchant that the customers have paid, and require the merchant to deliver the goods within the specified time;

(4)Merchant receives the notification, and delivers the goods in accordance with the order;

(5)If customers receive the goods, and they notify TTP, turn to (6); if customers reject the goods, and then they notify the TTP, turn to (7).

(6)TTP receives the notification of receipt, and their accounts transfer to the merchant's account, the transaction is completed (transaction successful);

(7)TTP makes their accounts returned to the customer's account, the transaction is completed (transaction failed).

B Model Checking and SPIN

Modal checking is a technique that relies on building a finite model of a system and checking that a desired property holds in the model or not. As specified in [8], Model checking has been used primarily in hardware and protocol verification. Currently it employed in software system also. Two approaches for model checking are used. First is temporal model checking in which specifications are expressed in a temporal logic and systems are modeled as finite transition system. In the second approach the specification is given as an automaton and the system is also modeled as automaton, and is compared to the specification to determine whether its behavior conforms to the specification [16].

There is a wide variety of model checking tools available, such as the SPIN [6], the NuSMV2 [9], Java Pathfinder [10], FDR and the MARIA [11]. Among them, the SPIN model checker represents the most popular one that provides a friendly user interface and accepts model specifications written in PROMELA (PROcess MEta LAnguage) [6, 12]. PROMELA is a language for building verification models that represent an abstract of a system, which contains only those aspects that are relevant to the properties one wants to verify [12]. A PROMELA program consists of processes, message channels, and variables. Processes are defined globally; while message channels and variables can be declared either globally or locally within a process. Processes are used to specify system behaviors, and channels and global variables are used to define the environment in which the processes run. Examples and further details about the PROMELA language can be found in references [6, 12].

There are two basic ways to use the SPIN model checking tool in system verification [12]. The first approach is to use the tool to construct verification models that can be shown to have all the required system properties. Such verification models can serve as specification models or high-level design models of a system to be implemented. The second approach is to start from an existing system, and based on the existing system, we build verification models that preserve the system behaviors to be verified [13].

The two main types of temporal logic used in model checking are Computation Tree Logic (CTL) and Linear Temporal Logic (LTL). CTL is a branching time logic that is most suitable for applications in hardware verification; while LTL is a linear time logic that is typically used for applications in software verification [12]. The SPIN model checker supports specification of system properties using LTL, which has been proven to have good expressivity and more natural language like statements for verification [14, 15]. LTL consists of only a few logic operators, such as "[]" (always), "<>" (eventually), "U" (until), "W" (unless, or weak until) and "O" (next). Combining with Boolean operators, i.e., "&&" (and), "||" (or), "!" (Negation), "→" (logical implication) and "↔" (logical equivalence), LTL is

capable of describing many key properties of a concurrent software system.

III. MODELING THE INTERNET PAYMENT SYSTEM

In this section we describe main features of the Internet Payment Systems which incorporates have been presented in last sections, and sketch how they can be specified in PROMELA.

Formal modeling is the first and crucial step in model checking. In the formal modeling process, we should ignore the participants which are independent of the desired characteristics of the system. In order to get an accurate model of the Internet Payment System, here we only pay attention to the three objects, the customer, the merchants and the trusted third party. We overlooked some objects such as the participants for storage or transport of goods, so it can simplify the system modeling process, and also can reduce the time and space overhead in the validation process.

The internet payment transaction protocol involves three participants: the consumer (C), the merchant (M) and the trusted third party (TTP). We use the notation "$X \Rightarrow Y$ *Message*" to indicate that *X* sends the message to *Y*. The basic protocol consists of the following messages:

1. $C \Rightarrow M$ *Buy*
2. $M \Rightarrow C$ *BuyOK or BuyNOK*
3. $C \Rightarrow TTP$ *Pay or Deny*
4. $TTP \Rightarrow M$ *Pay or Deny*
5. $M \Rightarrow TTP$ *Sent*
6. $TTP \Rightarrow C$ *Sent*
7. $C \Rightarrow TTP$ *Confirm or Back*
8. $TTP \Rightarrow M$ *Moneytos*
9. $TTP \Rightarrow M$ *Back*
10. $M \Rightarrow TTP$ *Backconf*
11. $TTP \Rightarrow C$ *Moneybackc*

The massage flow is shown in Figure 1 below.

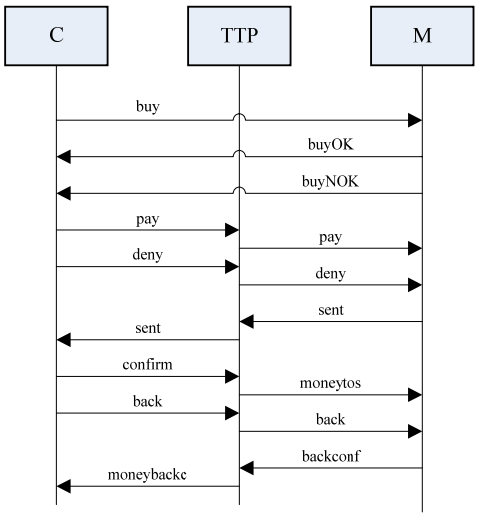

Figure 1. Massage flow in the Internet Payment Systems

Constructing a model for a protocol in PROMELA requires a previous abstraction process of the original source code. Usually, this process eliminates details that are not necessary for debugging purposes. Therefore, models will be as small as possible making sure that they represent the exact details needed for the properties to be analyzed.

Extended Finite State Machine (EFSM) has been the underlying model as formal description for the communications protocol. EFSM model is extended with the finite state machine (FSM) model. Compared with FSM, there are environmental variables and the migration of pre-conditions in EFSM. So EFSM model has a stronger ability to describe the dynamic behavior of the system. For these reasons, we use EFSM to model the process, which is formula in the area of model checking, and also can be described in PROMELA easily.

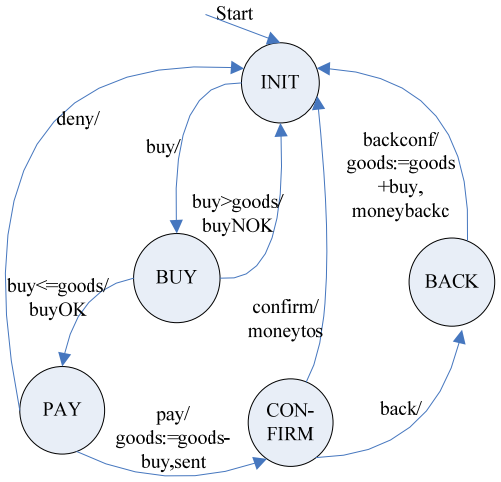

Figure 2. EFSM of the Internet Payment Systems

Definition: EFSM ***M*** is defined as the tuple $(S,s_0,V,M_V,P,M_P,I,O,T)$, in which:

S is a finite set of states, s_0 is the initial state, $s_0 \in S$;

V is the finite set of the internal variable (environment variable), and the range of the internal variable is D_V;

M_V is the set of the initial (or default) value of variables in *V* , in which any element can be expressed as a tuple (s, v), $s \in S$, $v \in D_V$;

P is the input and output parameters;

M_P is the set of the initial (or default) value of variables in *P*, in which any element can be expressed as a tuple (p, u), $p \in I \cup O$, $u \in Dp$, *Dp* is the range of the input and output parameters;

I is a set of the input symbols;

O is a set of the output symbols;

T is a finite set of state transition.

A state transition *t* ($t \in T$) is defined as the tuple (s,x,y,g_P,g_E,op,e), where:

s and *t* are the start (head) state and the end (tail) state;

g_P is the input and output conditions to determine;

g_E is the conditions to determine of the variable required for migration;

x and *y* are the input and output symbols;

op is output operations.

Figure 2 shows the EFSM model of the Internet Payment System, containing 5 states and 8 state transitions. The label "*buy<=goods/buyOK*" means that, when the input symbol of the state *BUY* satisfying "*buy<=goods*", the state will be converted to the state *PAY*, and it will output the symbol *buyOK*. The label "*pay/goods:=goods-buy, sent*" means that, when the input symbol is *pay*, the state *PAY* will be converted to the state *CONFIRM*, and it perform the operation "*goods:=goods-buy*" and output the symbol *sent*.

According to the analysis of message flow above, we have abstracted three processes, named *Customer*, *Seller* (Merchant), and *Netpay* (abstract of the TTP and the back). The fragment of PRMELA code that represents EFSM from Figure 2 is presented in Figure 3 as the Customer Process. In this fragment, each label like "*cus_init"* represents each state of the process. Notation "∷" denotes the possible nondeterministic choices of further execution options inside bodies of if and do operations. The option can be selected if its guard (the first statement followed after "∷") is enabled, that is executable. If more then one option is enabled, one will be selected at random. Notation "*goto"* means the process turn from one state to another state. Notation "*customer_seller!buy"* means sending value from variable *buy* to channel *customer_seller*. The variable *buy* defined for the number of goods bought by the customer. The similar situation is shown in Figure 4 for the seller process. The variable *goods* defined for the number of goods left in the seller. Notation "*customer_seller?buy"* means receiving a message from channel *customer_seller* into variable *buy*. The Netpay process is similar as the processes discussed above, we would not show the fragment in this paper. More details about PROMELA semantics can be found in [12].

```
proctype customer(int i)
{
  buy=B;
cus_init:
   do
          ::customer_seller!buy -> goto cus_pay
   od;
 cus_pay:
          ……
}
```

Figure 3. Fragment of Customer Process in PROMELA

```
active proctype seller()
{
 goods=50;
 sel_init:
 if
  ::customer_seller?buy ->
    if
      ::buy <= goods -> seller_customer!buyOK;
                                goto sel_wait
      ::buy >  goods -> seller_customer!buyNOK;
                            goto sel_init
    fi;
 fi;
 sel_wait:
          ……
}
```

Figure 4. Fragment of Seller Process in PROMELA

```
……
chan customer_netpay=[0] of {mtype};
chan customer_seller=[0] of {int};
chan seller_netpay=[0] of {mtype};
chan netpay_customer=[0] of {mtype};
chan seller_customer=[0] of {mtype};
chan netpay_seller=[0] of {mtype};
……
```

Figure 5. Fragment of the channels' definition in PROMELA

Channels are used to model data flow between processes and can be either globally scoped or locally scoped within a single process. Channels can have a predetermined storage capacity. When the channel capacity has been reached, additional messages sent to the channel will be dropped. The fragment presented in Figure 5 shows how to define the channels.

In Figure 3 and 4, there is a different point between them. In Figure 4, there is a notation *active* before the notation *proctype*. It means the process is alive when the program begins running. Because there can be several Customer processes, so we use this *init* process to run these Customer processes. The init process is shown in Figure 6 below.

```
init {
    int i;
    i = 1;
    do
        :: (i <=5) ->run customer(i); i++

        :: else ->
                    break
    od;
    }
```

Figure 6. Fragment of init process in PROMELA

IV. Description of the Model and Properties

In the last section, the Internet Payment System modeling in PROMELA has been completed. In the modeling process, we have done lots of abstraction to prevent the state explosion. The main purpose of this paper is to verify the process of internet payment transactions for any errors, so the model does not reflect the identity, password authentication and other security aspects of the process, these parts will also be the focus of our future research. Although it has been abstracted, this model still can describe the actual process of the internet transactions. The more details we will introduce are as following.

A Partial Return.

The actual business is to buy, return, and partial return. But in the model, it is given that only two types of business processes, to buy and return. Figure 7 shows the two different behaviors of the customers. In fact, partial return can be seen as made of two customers. One customer is to buy and not return, and the other is to all return. Thus, our model can represent all kinds of behaviors of customers.

In more detail, one customer buys *i*, the number of goods, returns *j* to the seller. Ii is equivalent to that, there are two abstract customers in the model, one buys *(i-j)*,

and the other buys *j* , returns all *j* back to the seller. The states of the two customers actually represent the customer who is in the case of the partial return.

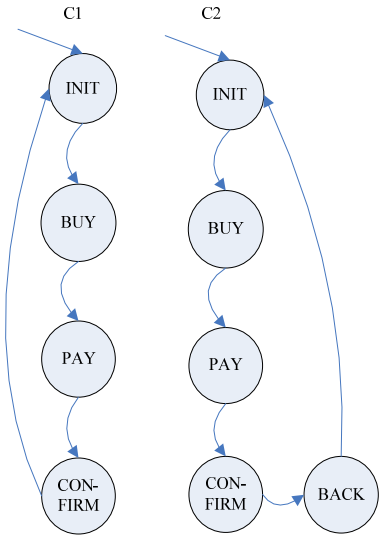

Figure 7. Two different behaviors of customers

B Several Cases of Different Values of Variables

In the model, there are three main variables represent the different meanings. The variable *i* means the number of customers. The variable *buy* means the number of goods bought by each customer. And the variable *goods* means the inventory of merchant. The three variables given different initial values, there will be several different situations, which represent the different actual payment business.

There are two major types of cases of the variable *i*. When it is *i<=1* in the loop in Figure 6, there is only one customer process running. When *i* <= any number larger than 1, there will several concurrent customer processes running together. If there are concurrent processes running, the Seller process will control each customer to turn correct state. Specific control process can be seen in the fragment of Seller process in Figure 4, and the detail of the next states is shown in Figure 8.

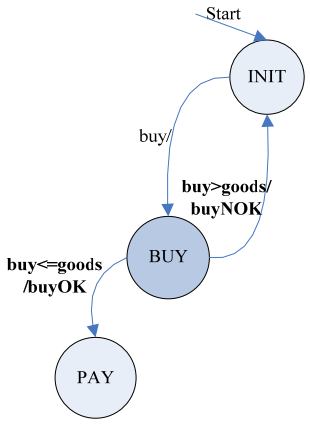

Figure 8. Different situations of the state BUY

In Figure 8, we will see that the other two variables *buy* and *goods* are compared in the process. Also it represents two different situations. If it is the situation of *"buy<=goods"*, it means the customer can buy the goods, as there are enough inventories of the seller. If it is the situation of *"buy>goods"*, it means there are not enough inventories. The process will turn back to the state INIT. So, if these two variables are defined by the initial values to meet the condition *"buy>goods"*, there will be some of the states that could never been reached. We will discuss these cases in more detail with experiments in Section 5.

C Properties Description

As explained in Section 2, SPIN supports two main kind of analysis for the modeled protocols. The first one consists of checking deadlocks and other safety properties by generating the execution paths in the model. The second kind of analysis consists of checking temporal properties specified with temporal logic. Here we describe correctness criteria we are interested in, and show how they can be defined in SPIN. In order to formalize both desired and undesired properties of the Internet Payment Systems, we use LTL (linear temporal logic notation), which has been explained in Section 2. LTL allows expressing temporal properties we expect the system behavior will conform to during the system lifetime. Such properties can be seen as a part of requirement specification. Expression of properties in the formal LTL notation gives both an unambiguous presentation of expected system behavior and possibility to verify whether the system model conforms to the requirements. LTL formulae can express both safety and aliveness properties, and are effectively supported in SPIN.

Examples of typical requirements to the internet payment behavior can be formulated in plain English as following. For customers, the customers need the support of the payment is guaranteed:

(1) Before receiving confirmation, the trusted third party will not pay the money to the merchant;

(2) IF customers are not satisfied with the goods, after the required return, the trusted third party will refund money to customers;

For the merchant, the business requires the support of the Internet Payment System guaranteed: After confirmation of receipt in customer, the trusted third-party party will pay the money to the merchant. That requires the support of the system:

(3) When transaction succeeds, the customers receive the goods, the merchant get money from the trusted third party.

(4) When transaction fails, the goods are returned to the merchant, and the money is back to the customers from the trusted third party.

According to section of the formal modeling and system security requirements, several formulas should be defined. The fragment is shown in Figure 9, in which, *p1* represents the customer confirms the transaction, *q1* means the TTP pays the money to the merchant, *p2* represents the TTP tells the merchant that the customer

want to return the goods, and *q2* means the TTP gives the money back to the customer.

```
……
#define p1 customer_to_seller == confirm
#define q1 netpay_to_seller == moneytos
#define p2 netpay_to_seller == back
#define q2 netpay_to_customer == moneybackc
……
```

Figure 9. Fragment of formulas defined in PROMELA

In this case the requirement expressing correct behavior can be expressed in LTL formula as: $<>(p1 \rightarrow q1) \&\& <>(p2 \rightarrow q2) \&\& <>!(p1 \&\& q2) \&\& <>!(p2 \&\& q1)$ Since SPIN analyzes all possible behaviors of the model including the case when the LTL formula above is false, the formula is false will always be discovered if it exists. The properties expressed in the form of linear temporal logic statements will be verified by applying model-checking system SPIN in the next section.

V. EXPERIMENTS

If a specification of model has been given in PROMELA, SPIN can search the whole state space of the model; it also can identify unreachable state or deadlock in model. In addition, SPIN can construct a verifier, which can check several claims on the execution of the model. We have established the model of the Internet Payment System in PROMELA, and analyzed the properties in last sections. These properties include the values of certain variables at certain points in the code and true statements that can be made about execution states (state properties) or the paths of execution (path properties). In this section, we present various kinds of verification that can be performed on a PROMELA model described in the sections above. Using SPIN and the PROMELA specification presented above, several properties of the execution of the model were verified. These properties were verified as part of several experiments described below. For each experiment, the size of the model constructed by SPIN, the time for verification were measured. The experiments were carried out on a 2.0GHz Pentium duel machine with 2048MB of memory.

A Experiment 1

First we have let SPIN perform a full state space search for invalid end states, which is SPIN's formalization of deadlock states, in case of 5 customers, *buy=2*, and *goods=100.* The results are shown in Figure 10. Here we give the meaning of each line in the results. The first line lists the version of SPIN used in the experiments. In line 2, the "+" represents the default simple algorithms is used, else, the "-" represents it is fully complied without simplification. We can add options "- DNOREDUCE" to achieve a fully compiled. Line 3 represents the type of search. In this case, it is the default type, that is all state-space search. The "-" in line 4 means it is not use "never" declare or LTL formulas. Line 5 indicates that the detection process does not violate the user-defined statements (assert). Line 6 indicates that the process does not detect the current user-defined acceptable cycle or no progress marked circle. The "+" in line 7 represents the correct end state, which means there is no deadlock. The following lines are easy to understand. We focus on the lines from 15 to 25. These lines represent the unreached states. Since the model of the Internet Payment System is an infinite loop, the process will never reach the end state.

```
(Spin Version 6.1.0 -- 4 May 2011)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (not selected)
        invalid end states      +

State-vector 100 byte, depth reached 3162, errors: 0
 14750894 states, stored
 32353853 states, matched
 47104747 transitions (= stored+matched)
     0 atomic steps
hash conflicts:  26864854 (resolved)

 1616.172  memory usage (Mbyte)

unreached in proctype customer
        NETPAY04.pml:89, state 38, "-end-"
        (1 of 38 states)
unreached in proctype seller
        NETPAY04.pml:139, state 43, "-end-"
        (1 of 43 states)
unreached in proctype netpay
        NETPAY04.pml:176, state 35, "-end-"
        (1 of 35 states)
unreached in init
        (0 of 10 states)

pan: elapsed time 51.8 seconds
pan: rate 284711.33 states/second
```

Figure 10. Results of verification of 5 customers

TABLE I.
RESULTS OF VERIFICATION OF 1-5 CUSTOMERS

Number of Customers	Elapsed time	Memory usage	States stored	States matched	Transitions	Depth
1	0.02s	3.418Mb	10952	5221	16173	1087
2	0.11s	8.105Mb	61358	69959	131317	1198
3	0.45s	24.902Mb	244672	269288	513960	1264
4	6.02s	210.832Mb	2023217	3361364	5384581	1426
5	51.8s	1616.172Mb	14750894	32353853	47104747	3162

B Experiment 2

We have let SPIN perform a full state space search in cases of 1-5 customers, and the variable *buy=2*, *goods=100*. The results are summarized in Table I. The exponential increase in number of states, memory usage and verification time, seems to be not manageable when checking more than 5 customers. In case of 6 customers, the available physical memory was insufficient.

Since checking more than two customers is superfluous and violates the requirement that the verification model be the minimum sufficient model to perform the verification successfully. In Figure 11, we can see the memory usage is in the exponential growth with the number of customers increasing. We do not gain in verification power by checking more than 5 customers. In our future research we will search for a better modeling of loops that will minimize the state explosion that has been revealed by our experiments.

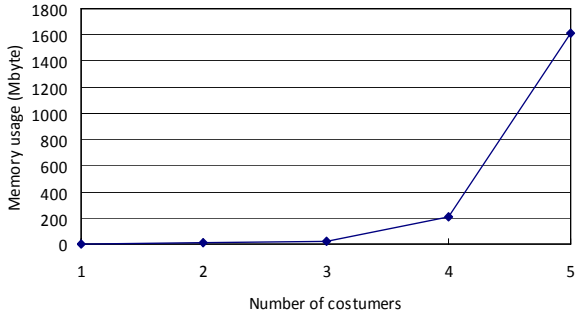

Figure 11. Memory usage of 1-5 customers

C Experiment 3

In the experiments, we define the variable as a special case, in which we define the variable *buy*=3 and *goods*=2. This means that in the initial state, the determination is consistent with "*buy>goods*". We have let SPIN perform a full state space search in this case. The results are shown in Figure 12. The meaning of the first few lines in the results has been introduced in above cases. Compared with the results in Figure 10, we will find the differences mainly in "*unreached in proctype*". In other words, many states are unreachable in this case.

Because the two variables are defined by the initial values to meet the condition "*buy>goods*", all customers will not succeed to buy the goods. As shown in Figure 8, all customers will get the massage "*buyNOK*", which means they cannot buy the goods. So the situation "*seller_customer!buyOK*" will never arise. All the customers in the state *BUY* will turn back to the state *INIT*, and the states behind the state *BUY* would never been reached. For this reason, it is acceptable that there are so many states unreachable in the results shown in Figure 12.

In the experiments, we also have tried to simulate the actual situation as more as possible, and add the property which we will verify into the process. In the actual internet payment, the quantity of goods bought by each customer is not the same. Although we can not simulate the random number of goods bought by each customer, we can define the variable *buy=i*, approximating the actual situation. Thus, the process can simulate different numbers of goods for each customer. Several interesting liveness claims can be made about the Internet Payment System. We have described the properties in LTL formula in last section. The experiment shows that the verification of the model of the Internet Payment System that does not contain any loops can be done very effectively. The results of these experiments show that there is no error in the design of the Internet Payment System.

```
(Spin Version 6.1.0 -- 4 May 2011)
        + Partial Order Reduction
.....
State-vector 76 byte, depth reached 11, errors: 0
   13 states, stored
    3 states, matched
   16 transitions (= stored+matched)
    0 atomic steps
hash conflicts:      0 (resolved)

 2.539      memory usage (Mbyte)

unreached in proctype customer
    NETPAY04.pml:73, state 28, "customer_netpay!confirm"
        .....
        (3 of 37 states)
unreached in proctype seller
    NETPAY04.pml:99, state 4, "seller_customer!buyOK"
        .....
        (10 of 43 states)
unreached in proctype netpay
    NETPAY04.pml:145, state 2, "netpay_seller!pay"
        .....
        (11 of 33 states)
unreached in init
        (0 of 10 states)

pan: elapsed time 0 seconds
```

Figure 12. Results of verification of the buy>goods condition

If we have detected any situation which do not meet the properties in the other model of protocol, SPIN can display these paths. We call these paths as counter-examples. According to the counterexample generated by SPIN, developers have the opportunity to understand wrong business process behavior, to locate errors and to effect right changes for correcting business process design. Then, the modified design is again submitted to SPIN for verification. This methodology determines a gradual correction and refinement of business process models, before it is definitely implemented.

VI. Related Work

Research on model checking has been conducted based on an underlying logical representation. Many tools have been built to model checking FSM- or EFSM- based systems. Tools for Model checking such as SMV [9], SPIN [6] and FDR have been used by researchers and industrialists to find bugs in circuit designs, floating point standards, cache coherence protocols for multiprocessors.

Recently model checking has also been applied to check software systems and protocols. Parts of our work refer to model checking fault tolerance with respect to the Internet Payment System guarantees.

The vast majority of Internet Payment Systems are online systems that perform either. A detailed description of the forenamed types of Internet Payment Systems is given in [17, 18, 19] .In Reference [20], the authors provide a thorough treatment of the most fundamental security requirements, as well as a complete source of references. In Reference [21], the authors point out the need for custom-made payment systems, which provide payment services that are extended beyond the traditional bilateral transaction model. There are also many papers introducing the development of the Internet Payment Systems, such as Reference [28]. The system discussed in our article is based on the Internet Payment Systems introduced in all the papers above.

The authors of [23] point out the need for a systematic treatment of the correctness properties required in Internet Payment Systems. Reference [24] is finding that e-Business managers, developers, and auditors require robust tools to assure users that Internet Payment Systems are secure and reliable. Designing and implementing highly secure and reliable e-processes is challenging and requires adherence to several specific criteria to be effective. The authors of [22] use a model checking tool to examine the non-security characteristics of e-processes to verify the money and goods atomicity properties of two online payment processes, which are Digicash [25] and NetBill [26]. This seminal work demonstrated how to model the Internet Payment Systems and their properties of interest in a process algebra language, CSP, which is the language used in FDR [27].The author of [1] gives a roadmap to electronic payment transaction guarantees and a Colored Petri NET model checking approach. The paper uses the model checker based Colored Petri Net.

Our work builds on the insights in the above work in model checking methods which are using SPIN and specified declaratively by EFSM. There are also many papers in the area model checking with SPIN, but not for the Internet Payment Systems, such as Reference [29, 31, and 32]. These papers also bring a lot of learning to our approach.

Our approach improves on the internet payment in three directions. First, we provide a model based EFSM for the Internet Payment Systems in PROMELA. As a result, the verification procedure can detect interactions of each behavior of online payments. Second, as the important part of our modeling methodology, we translate the Internet Payment System into a simpler model that nevertheless preserves all the essential behavior to be verified. Third, we provide initial results on the actual verification of the Internet Payment System using SPIN. The result of our work is a complete procedure for the modeling and verification of the Internet Payment System.

VII. Conclusions and Future Work

In this paper, we introduced a model checking approach to verify the Internet Payment Systems. First we analyzed the general Internet Payment Systems, including the internet payment protocol, the participants of the payment transactions, and the massage flow in the system. We proposed an EFSM model, and translated the model in PROMELA. We also summarized a set of LTL formulas that can guarantee the reliability of the transactions. And then we did some experiments, which can prove that our model can simulate the actual transactions. Also the initial results on the verification of the Internet Payment System using SPIN were provided. Our approach can be easily extended to support model checking debugging new design of the Internet Payment Systems using the SPIN tool. The designers may simulate their active applications with our method.

For our future work, we will try to combine our approach with more details of the Internet Payment Systems, such as password authentication and other security-related properties. Also in our future research, we will search for a better modeling of loops, which will minimize the state explosion revealed by our experiments.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61070039, the Fund for the Doctoral Research of Shandong Academy of Sciences under Grant No. 2010-12, and the Outstanding Young and Middle-aged Scholars Foundation of Shandong Province of China under Contract No. BS2011DX033.

References

[1] P. Katsaros, “A roadmap to electronic payment transaction guarantees and a Colored Petri Net model checking approach,” *Information and Software Technology, Elsevier*, vol.51, pp. 235–257, 2009.

[2] M. M. Gallardo, J. Martinez and P. Merino, “Model checking active networks with SPIN,” *Computer Communications, Elsevier*, vol.28, pp. 609–622, 2005.

[3] E.M. Clarke, E.A. Emerson and A.P. Sistla, “Automatic verification of finite-state concurrent systems using temporal logic specifications,” *ACM Trans. on Programming Languages and Systems*, vol.8 (2), pp.244–263, Apr. 1986.

[4] E. Clarke, O. Grumberg and D. Peled, *Model Checking*, MIT Press, Cambridge, 2000.

[5] G.J. Holzmann, *Design and Validation of Comp. Protocols*, Prentice-Hall, Englewood Cliffs, NJ, 1991

[6] G.J. Holzmann, “The model checker SPIN,” *IEEE Transactions on SE,* vol.23 (5), pp. 279–295, 1997.

[7] On-the-fly LTL model checking with SPIN. http://spinroot.com/spin/whatispin.html

[8] E. M. Clark and J. M. Wing, “Formal methods: State of the art and future directions,” *ACM Computing Surveys*, Vol. 28(4), pp. 1–18, 1996.

[9] A. Cimatti, E. M. Clarke, E. Giunchiglia, et al. “NuSMV2: An OpenSource Tool for Symbolic Model Checking,” *Proceeding of International Conference on Computer-Aided Verification (CAV2002)*, Copenhagen, Denmark, July 2002.

[10] K. Havelund. “Java PathFinder: A Translator from Java to PROMELA,” *Proceedings of the 5th and 6th International*

SPIN Workshops on Theoretical and Practical Aspects of SPIN Model Checking, Springer-Verlag, pp. 152, 1999.
[11] M. Makela, "Maria: Modular Reachability Analyser for Algebraic System," *Application and Theory of Petri Nets 2002, 23rd International Conference, ICATPN 2002,* Vol. 2360, pp. 434–444, June 2002.
[12] G. J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*, Addison Wesley, 2004.
[13] H. Xu and Y-T. Cheng, "Model Checking Bidding Behaviors in Internet Concurrent Auctions," *International Journal of Computer Systems Science & Engineering (IJCSSE)*, Vol. 22, No. 4, pp. 179–191, July 2007.
[14] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: Specification,* Springer-Verlag, 1992.
[15] E. M. Clarke, O. Grumberg, and K. Hamaguchi, "Another Look at LTL Model Checking," *Formal Methods in System Design*, Vol. 10, pp. 47-71, February 1997.
[16] R. Shaikh and S. Devane, "Formal verification of payment protocol using AVISPA," *International Journal for Infonomics*, Vol.3, Issue 3, September 2010.
[17] H.M. Deitel, P.J. Deitel and T.R. Nieto, *e-Business & e-Commerce: How to Program*, Prentice Hall, 2001.
[18] A. Pombortsis and A. Tsoulfas, *An Introduction to Electronic Commerce*, Tziolas Publishers, 2002.
[19] L. Ferreira and C. Dahab, "A scheme for analyzing electronic payment systems," *Proceedings of the 14th Annual Computer Security Applications Conference,* IEEE Computer Society, pp.137–146, 1998.
[20] N. Asokan, P. Janson, M. Steiner, et al. "State of the art in electronic payment systems," *IEEE Computer*, vol.30 (9), pp. 28–35, 1997.
[21] H. Schuldt, A. Popovici and H.J. Schek, "Automatic generation of reliable e-commerce payment processes," *Proceedings of the First International Conference on Web Information Systems Engineering (WISE00)*, IEEE Computer Society, vol. 1, pp. 434–441, 2000.
[22] N. Heintze, J. Tygar, J. Wing, et al. "Model checking electronic commerce protocols," *Proceedings of the Second USENIX Work-shop in Electronic Commerce,* Oakland, CA, USENIX Association, California, pp. 146–164, 1996.
[23] B. Pfitzmann and M. Waidner, "Properties of payment systems – General definition sketch and classification," *Research Report RZ 2823(#90126)*, IBM Research Division, May 1996.
[24] I. Ray, "Failure analysis of an e-commerce protocol using model checking," *Proceedings of the Second International Work-shop on Advanced Issues of e-Commerce and Web-based Information Systems*, Milpitas, CA, 2000.
[25] D. Chaum, A. Fiat and M. Naor, "Untraceable electronic cash," *Advances in Cryptology, CRYPTO '88 Proceedings,* 1990.
[26] B. Cox, J. Tygar and M. Sirbu, "Netbill security and transaction protocol," *Proceedings of the First USENIX Workshop in Electronic Commerce,* July 1995.
[27] G. Lowe, "Breaking and fixing the needham-schroeder publickey protocol using FDR," *Tools and Algorithms for the Construction and Analysis of Systems: Second International Work-shop*, TACAS, March 1996.
[28] Z. Duric, O. Maric and D. Gasevic, "Internet Payment System: A new payment system for internet transactions," *Journal of Universal Computer Science,* vol. 13(4), pp.479-503, 2007.
[29] R. Zeng and X. He, "Analyzing a formal specification of mondex using model checking," *Lecture Notes in Computer Science,* Springer-Verlag, Berlin Heidelberg, vol.6255, pp. 214-229, 2010.
[30] H. Cao, S. Ying and D. Du, "Towards model-based verification of BPEL with model checking," *Proceedings of the sixth IEEE international conference on computer and information technology (CIT06), IEEE,* 2006.
[31] J. Bentahar, J. Meyer and W. Wan, "Model checking communicative agent-based systems," *Knowledge-Based Systems,* Elsevier, vol.22, pp.142–159, 2009.
[32] M. H. Beek, M. Massink, D. Latella, et al. "Model checking Publish/Subscribe notification for thinkteam," *Electronic Notes in Theoretical Computer Science*, Elsevier, vol. 133, pp.275–294, 2005.

Wei Zhang was born in Tai'an, China, in 1983. He received the B.S. degree in Mechanics from Zhejiang University in 2004 and the M.S. degree in Computer Software and Theory from Liaoning University in 2008. He works in Shandong Computer Science Center, where his research interests are in Software Testing, Formal Verification, test sequence generation methods, and so on.