Analysis of an Electronic Payment System Using Simulation

Otto Poszet, Ovidiu Novac, Stefan Vári-Kakas

Department of Computer Science, Faculty of Electrical Engineering and Information Technology, University of Oradea
Univeritatii Str., nr.1., Oradea, Romania
E-mail: poszet@uoradea.ro; ovnovac@uoradea.ro; vari@uoradea.ro

Abstract: In this paper we propose a new, fault tolerant on-line EPS. After the analysis of all the possible failure scenarios, for each case we propose solutions based on roll-back and recovery mechanism (repetitions, time-out, acknowledge messages, logs, resending, synchronizations). Based on the Monte-Carlo method, we realized a simulation of system behavior, in order to obtain the EPS's correct service rate vs. the messages loss rate between client-bank and shop-bank. Based on this simulation we concluded the increase of the success rate of transactions in the fault tolerant case vs. the un-tolerant case.

Keywords: Electronic Payment System, data security, fault tolerance, Monte-Carlo simulation

1 Introduction

The main difference between an on-line and an off-line Electronic Payment System (EPS) is that the payment protocol in the case of the on-line systems is monitored, checked and authorized by a Trusted Third Party (e.g. the bank). In the off-line systems, the payment protocol is executed only between the client and the shop, without a Trusted Third Party. So, this kind of EPS-s can guarantee more freedom for clients, as the on-line EPS-s, but their main disadvantage is that the fraud detection can be made only after the payment, in the deposit protocol. This is the reason, why the on-line payment systems are more often used, as the off-line systems. We can say that in the on-line EPS-s is ensured the preventive integrity and not only the degradation integrity (off-line case, increased security).

There are a lot of on-line systems implemented, and the main goal in development was to ensure security. Their study demonstrated that the real systems have very few embedded fault tolerance mechanisms, or they have not at all. We propose a new on-line EPS, which is similar to the existing EPS-s (e.g. Visa, PayPal), but provides fault tolerant mechanisms.

2 Architecture of the Proposed, Fault-Tolerant, On-Line EPS

The proposed, on-line EPS uses transactions between three kinds of entities: clients (payers), electronic shops (payees) and the bank (Trusted Third Party). The architecture of the proposed system is shown in Figure 1.

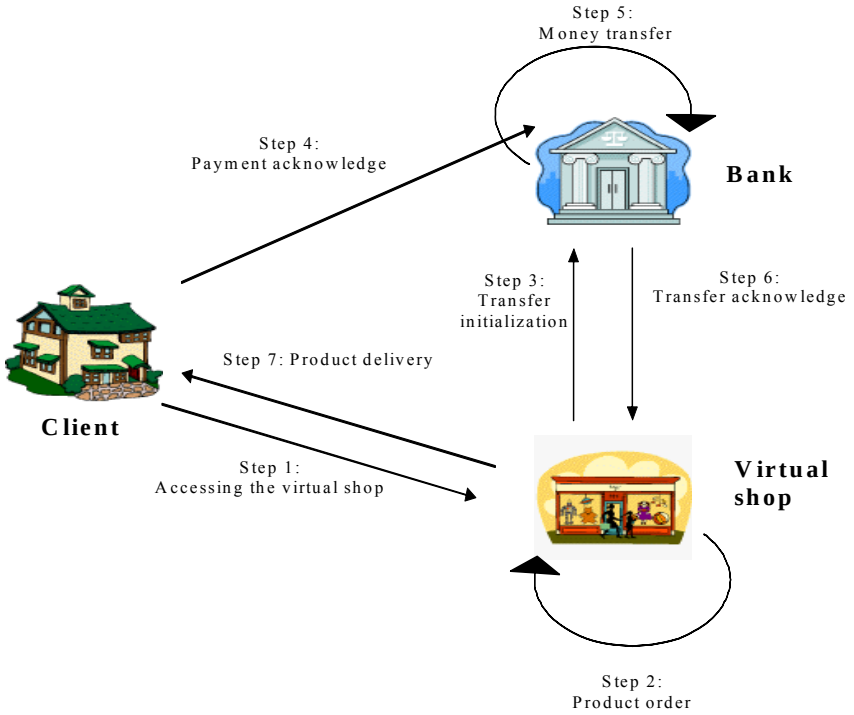

Figure 1
The architecture of the proposed, *on-line* EPS

The chronological order of the events in the case of an on-line shopping is:

1 The client, using a web browser, accesses the authorized site of the bank, section 'accredited virtual shops', and chooses a shop, using the link obtained from the bank.

2 The client selects some products or services from the shop, using his shopping cart, and launches the payment process.

3 The shop collects some information for the product delivery (e.g. address), generates a transaction's ID, saves all the data in a database, and redirects the client's browser to the bank site. The shop sends (encrypted) to the bank the transaction's ID, the amount of money and the shop's name to lunch a payment.

4 The bank asks the client to fill out a secure form for authentication: name, credit card number, date of issue, expiry date and password.

5 Based on this data, the bank verifies the identity of the client, the validity of the card, the sum of money in the account, and asks the client to confirm the money transfer to the shop's account.

6 If the bank receives the acknowledge from the client, the bank transfers the requested and confirmed amount of money in the account of the shop, saves the transaction in his data base, and sends an acknowledge to the shop, using a message signed digitally by the bank with the bank's secret key.

7 The shop delivers the paid product or service to the client.

8 The shop will save the received acknowledge message signed by the bank, and this message can be used later as a confirmation of the payment.

9 The client can verify every time his personal payment archive accessing the bank web site, and using his login name and password.

The proposed system ensures anonymity, because the client must not give his real personal data to the shop, for example his name, or his credit card number. The client launches the payment using the bank and only the bank can identify him. The shop doesn't sends to the bank all the detailed shopping information (e.g. the ordered product), only the transaction's ID and the amount of the money. So, the shop will not know the client real identity, and the bank will not know what the client buys. The shop knows only a delivery address, and the bank knows the shop and the amount of money used by the client.

3 Security and Fault-Tolerant Measures

To establish the needed security and fault tolerant measures, we analyses in chronological order the events that can take place in the system, and so we could identify the vulnerable points of the system.

To ensure security, the main proposed solutions are:

1 To avoid the shop's unwanted access to the client's confidential data, our proposed solution uses the redirection of the client identification and the payment transaction to the bank's authorized server.

2 For mutual identifications between client-bank and shop-bank we use a new identification protocol presented in [5].

3 The connections between shop-bank and client-bank are encrypted (Open SSL).

To ensure fault tolerance, we analyses all the possible failure scenarios and for each case we developed solutions based on roll-back and recovery mechanism (repetitions, time-out, acknowledge messages, logs, resending, and synchronizations):

1 If the web page of the bank does not load, the possible reasons can be the shop server or the bank server crash, or a failure in internet connection. The proposed solution is the canceling of the transaction at the shop and at the client, and the repetition of the process by the client a little bit later.

2 If the bank web site is loaded but the payment request between shop and the bank is lost or is late (step 3), we uses an acknowledge message from the bank to shop. If the shop doesn't receive this acknowledge after an amount of time (time-out), the shop will resend the message, repeating this process a fixed number of times. For the client the problem can be solved by using a transaction's history (paid or not-paid requests), and this history can be refreshed and viewed to see all the updates in transactions.

3 Arrived to the bank site, the client may refuse the payment of the products, or the acknowledge message from the client to bank can be lost (step 4) or the client may not have enough money in his account. In this case the bank must cancel the transaction after a time-out, and he must announce the shop and the client about the situation. The possible solution in this case is rolling back to the previous state, using a log.

4 If the money transfer in the bank doesn't realizes (from an account to another), the reason can be a bank server crash or insufficient money in account. In the first case the transaction will be repeated after the server restart, using a log. In the second case, the bank announces the shop and the client, cancels the transaction, and reestablishes the previous situation based on a log too.

5 If the client paid (the money transfer took place), but the payment acknowledge message between the bank and the shop is lost, this situation can be caused by a broken link between bank-shop, or by a server crash (bank or shop). The proposed solution is the use of an acknowledge message from the shop together with a time-out at the bank, and if this confirmation doesn't arrives, the bank can repeat the payment acknowledge message. Another solution is a periodic synchronization between the bank and the shop server, based on the transaction logs kept on both servers.

6 After the shop receives the payment acknowledge, he will send an acknowledgement about this to the client. The receiving or the losing of this message is not very important, because the client can verify the payment using the web site of the bank (in his account, in transactions history, payment checked), and because the products will be delivered in a time interval known by the client.

4 Test Results

Based on the Monte-Carlo method, we realized a simulation of system behavior, in order to obtain the EPS's correct service rate vs. the messages loss rate between client-bank and shop-bank. We defined the events 'success' and 'malfunction' regarding the service provided by the EPS system

$$Success = (payment \wedge delivery) \vee (not\_payment \wedge not\_delivery) \quad (1)$$

$$Malfunction = (payment \wedge not\_delivery) \vee (not\_payment \wedge delivery) \quad (2)$$

To fill out the truth table for 'SuccessU' (success in the un-tolerant case) and 'SuccessT' (success in the fault tolerant case) for the behavior of the whole system, we analyses the architecture shown in figure1, the definitions presented above and all the possible failure scenarios together with the proposed solutions. Table 1 shows all the possible failure situations in case of message loss between entities or erroneous message. We use four Boolean variables M1, M2, M3 and M4 (M=1 delivered message, M=0 lost or erroneous message) to describe the main links between entities: M1 – client-shop; M2 – shop-bank; M3 – client-bank; M4 – bank-shop.

Nr.	M1 Client-Shop Step 1+2	M2 Shop-Bank Step 3	M3 Client-Bank Step 4+5	M4 Bank-Shop Step 6+7	f1 SuccessU Un-tolerant	f2 SuccessT Tolerant	Comments
1	1	1	1	1	1	1	Payment and delivery
2	1	1	1	0	0	1	Payment, but the acknowledge from the bank doesn't arrives ⇒ undelivered product in the un-tolerant case
3	1	1	0	1	0	0	The acknowledge message for shopping is lost ⇒ not paid, but product delivery!
4	1	1	0	0	1	1	The shopping acknowledge is lost and no product delivery
5	1	0	1	1	0	1	The payment request is lost, but in the tolerant case this can be resend.
6	1	0	1	0	0	0	Payment without acknowledge and without product delivery
7	1	0	0	1	0	0	Not payment, but delivery ⇒ malfunction in both cases
8	1	0	0	0	1	1	Payment request without payment and delivery ⇒ success
9	0	1	1	1	1	1	The shopping request is lost, a human operator can resend it again
10	0	1	1	0	0	1	The shopping request can be send again. Payment, but no product delivery ⇒ malfunc. in the un-tolerant case
11	0	1	0	1	0	0	No payment, but product delivery⇒malfunction
12	0	1	0	0	1	1	Not payment, not delivery ⇒ success.
13	0	0	1	1	0	1	In the tolerant case we can resend the shopping request, and the shop can resend the payment request
14	0	0	1	0	0	0	Payment, not delivery ⇒malfunction
15	0	0	0	1	0	0	Not payment, delivery ⇒ malfunction
16	0	0	0	0	1	1	Not payment, not delivery ⇒ success

Table 1
Truth table for functions SuccessU and SuccessT

From Table 1, we can observe an interesting thing: the first variable (M1 – client-shop) doesn't influence the value of the two functions. This can be explained, because a human operator can be seen every time as a 'fault tolerant factor' – he

can repeat his last action, or he can resend a message, or he can refill a form. The variables M2 and M4 describe the same communication channel, only the direction of the messages differs. So, for the graphical representation of the test results we will use the axis Ox for the link shop-bank, axis Oy for client-shop and axis Oz for the function values (successU, successT). The algorithm computes the success rate in the two cases using the three variables M2, M3 and M4.

Our simulation program generates consecutively different loss rates for the following two links: shop-bank and client-bank (Δrate = 5% = 0.05), scanning all the existing possibilities. For each selected combination of type shop_bank_message_loss_rate ↔ client_bank_message_loss_rate, the program computes for a number of iterations = 500 times the values of functions f1=successU and f2=successT, respectively, using random numbers for simulation (Monte-Carlo method).

Test results are presented in Figures 2 and 3. It can be observed an approx. 10% increase of the transactions success rate in the fault tolerant case, principally caused by the successful treatment of the failures originated in the client side.

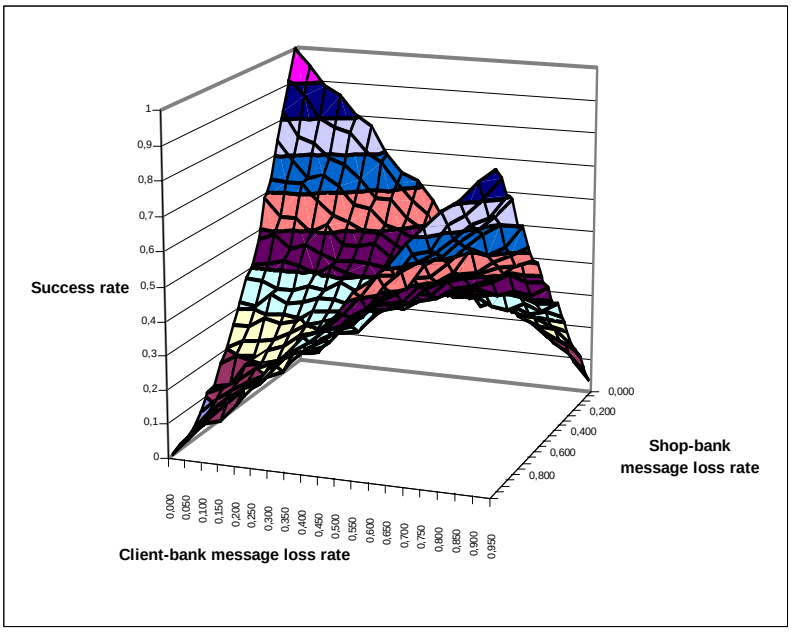

Figure 2
Success rate vs. message loss rate. The un-tolerant case.

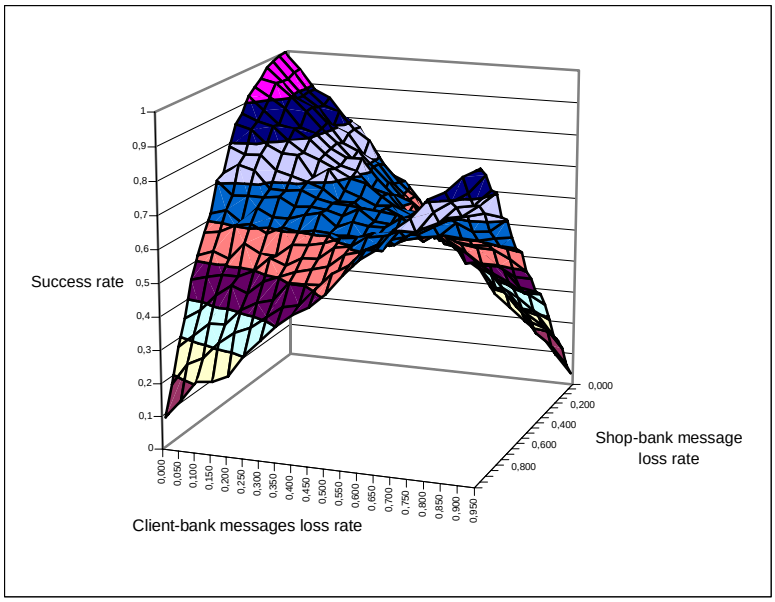

Figure 3
Success rate vs. message loss rate. The fault-tolerant case.

Conclusions

In this paper we proposed a new, fault tolerant, on-line EPS. The starting point for the analysis was the architecture of the system, and studying each transaction and each possible failure scenario, we discovered the vulnerable points of the system from the point of view of the security and fault tolerance. Based on this analysis we proposed security and fault tolerant measures to improve the system behavior. The result is an EPS more secure and more reliable. To demonstrate this conclusion, we computed the variation of the *EPS correct service* success rate in the *un-tolerant case* and in the *fault tolerant case* respectively, using the Monte-Carlo simulation. Based on this simulation we concluded the increase of the success rate of transactions in the fault tolerant case vs. the un-tolerant case.

A future research direction is the extension of the proposed system in order to include a set of banks in the system. In this case we need a new element: a payment processor for handling the inter-banking requests.

References

[1] Birmann, Kenneth P.: Reliable Distributed Systems. Technologies, Web Services and Applications, Springer Science and Business Media, Inc., 2005

[2] Jalote, P.: Fault Tolerance in Distributed Systems, Prentice Hall, 1994

[3] Pedragal-Martin, C., Ramamritham, K.: Guaranteeing Recoverability in Electronic Commerce, Proc. 3rd Int. Workshop WECWIS, 2001

[4] Pollard, J. M.: Monte Carlo Methods for Index Computation (mod p), Math. of Computations, Vol. 32, 1978, pp. 918-942

[5] Poszet O., Vári-Kakas S., Novac O., Ignat I.: Fault Tolerant Protocols Used in Electronic Payment Systems, Proceedings of the 6th International Conference on Renewable Sources and Environmental Electro-Technologies, Session Computer Science and Control Systems, University of Oradea, 2006, pp. 39-42

[6] Poszet O., Vári-Kakas S., Novac O., Drăgan H., Ignat I.: Efficiency of Identification Protocols in Electronic Payment Systems, Annals of the University of Oradea, Volume Electrotechnics, Session Computer Science and Control Systems, 2005, pp. 118-121

[7] Yaw, T.: Fault-Tolerance of e-Commerce: Transaction Protocols, BA Thesis, Boston College, 2003

[8] Rama, S., Vijayaraghavan, V.: Fault-Tolerance in E-commerce Web Servers, ECE Department, University of Wisconsin Madison, 2004