Developing A Scalable Payment Gateway using Scala

Sanmit Mallapur[1] Ketki Umbarkar[2] Forum Gala[3] Nikita Torawane[4] Prof. M.R.Patil[5]
[1,2,3,4]Student [5]Assistant Professor
[1,2,3,4,5]Department of Computer Engineering
[1,2,3,4,5]Smt. Kashibai Navale College of Engineering Pune 411041

Abstract— With the birth and expansion of the Internet, many electronic commerce systems have been developed and put into practice. In this paper, we introduce and develop an electronic commerce system prototype. The e-commerce system includes a merchant system and the payment gateway. The payment gateway helps in securing the transactions between the card issuing bank and the merchant system. The existing payment gateway systems are not scalable due to some factors. The proposed systems eliminate this bottleneck and make the entire system more robust. The above mentioned objective can be achieved with the help of Scala. Scala is a pure object-oriented language which uses immutability to handle concurrency. Our proposed protocol uses symmetric key operations which require low computational power and can be processed faster than asymmetric ones.

Key words: Scala, Payment Gateway, E-Commerce, Merchant System

I. Introduction

The accomplishment of payments is crucial part in business to customer electronic commerce (E-commerce) interaction since this phase does not only determine the success or failure but also requires sensitive information to be transferred. At the same time, several electronic payment systems are introduced. Every electronic commerce system generally includes three parts: data communication system, logistics system and electronic payment system [1]. Data communication system is an online procedure to establish a business. Logistics system is to deliver products. The electronic payment system is responsible for the payment and settlement. These electronic payment systems act as an intermediary between the merchant system and the card issuing bank. The card issuing bank is the entity which issues the debit card or credit card to the customer. There are different categories of payments available to the customer which are the debit card, the credit card, and net banking. A payment gateway is an e-commerce service that authorizes payments for e-businesses and online retailers [2]. Scalability and security are the main pillars of the payment gateway. Scalability will increase the number of users at the same time . The current payment gateways also charge a high amount of Setup fees and Transaction Discount Rate (TDR). The TDR fee is charged by the payment gateway to the merchant system for each and every transaction which can be costly for a new venture. The motivation behind developing a scalable payment gateway is the lack of scalability in the current technologies. The current payment gateways cannot handle a large amount of traffic concurrently. The leading payment gateways of India have a success rate of less than 50% [4]. Increasing the scalability of the system will increase the systems consistency and thus make it robust. By using Scala and the actor model, we are trying to boost the applications capacity to handle a large amount of data.

II. Background

A payment gateway is connected to all customers, merchants, and banks through the Internet and is responsible for the speed, consistency and the security of the transactions that take place. The payment gateway can be used to process the transaction through credit card, debit card as well as net banking. The basic architecture of the payment gateway involves three modules : The payment gateway itself, the merchant site and the card issuing bank. The entities in the transaction can be given as follows:

1) Customer’s End: The Customer is the person who wants to buy products online. Customer having an account in the system will have a unique id.
2) Merchant System: The Merchant System provides a portal for the customer to login and selects a product. The Merchant also forwards the customer details like customer id, etc. to the Payment Gateway. It will have a complete view of all the customer’s payments.
3) Payment Gateway: The Payment Gateway is the bridge between the Merchant System and the Card Issuing Bank. It forwards all the transaction request to the Bank for processing. It also sends a response from the Bank to the Merchant System to notify the customer of its transaction status.
4) Card Issuing Bank: The Bank authorizes the customer for the particular transaction. It plays a crucial role in the settlement of the authorized and captured transactions.
5) Acquiring Bank: The Bank that accepts debit card, or credit card transaction for a cardholder. It brings about the association between the payment gateway and the card issuing bank.

III. Architectural design

The process involves the user to feed the credit card details on either the payment gateway or the merchant website depending upon the nature of the payment gateway involved. Then the merchant site encrypts the transaction details and sends the data to the payment gateway using SSL encryption. The control is then transferred to the payment gateway. The payment gateway then allows the user to input the card details. These details are encrypted and sent to the acquiring bank. The acquiring bank brings about the settlement of funds between the merchant bank and the card issuing bank. To verify that a transaction received by the gateway is associated with the right user and originated from the merchant, the merchant attaches a message authentication code (MAC) of the transaction which is transmitted to the gateway along with the transaction details. The MAC generation algorithm is payment gateway specific. Gateway at its own will recompute and verify the MAC to make sure the data has not been tampered with and then proceeds with assessing the rest of the transaction.

After it verifies the transaction it redirects back to the merchant with the status of the transaction and the MAC of the status so the merchant can verify the request originated from the gateway. The sender (merchant) has a secret key (shared with the gateway) and uses a predetermined MAC algorithm to generate a MAC string which is included in a hidden form field while redirecting the user to the receiver (gateway).

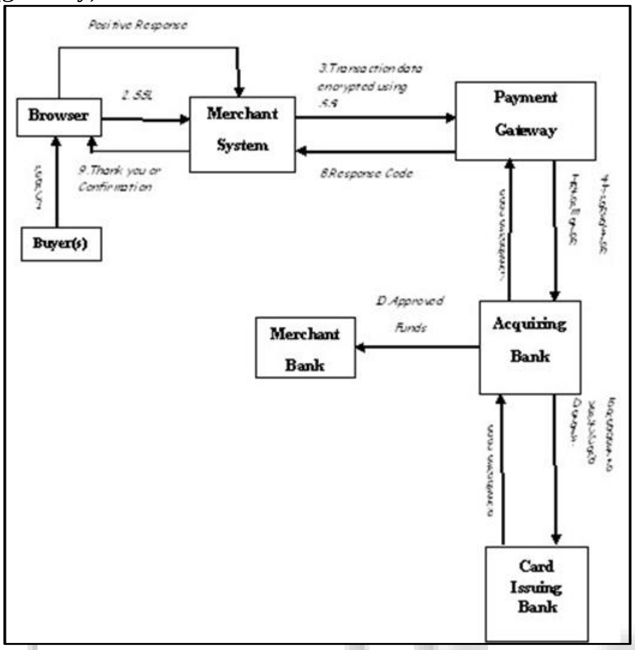

Fig. 1: Architecture Diagram

IV. PROPOSED METHODOLOGY

The proposed methodology uses Scala language and makes use of actor model for increasing its scalability and makes use of HMAC for security.

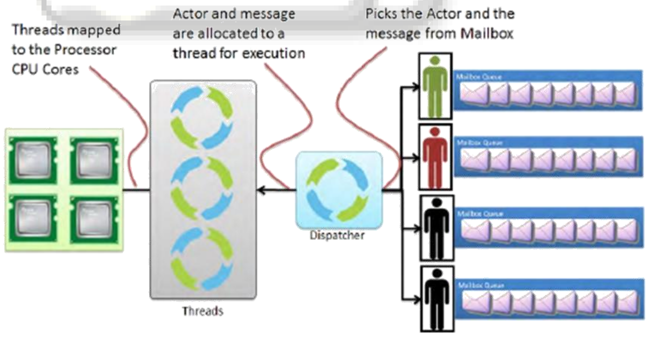
Threads mapped to the Processor CPU Cores
Actor and message are allocated to a thread for execution
Picks the Actor and the message from Mailbox
Threads
Dispatcher
Mailbox Queue

Fig. 2: Actor Model [5]

A. *Actor Model:*

In an actor-based system, everything is an actor, in much the same way that everything is an object in the object-oriented design. A key difference, though particularly relevant to our discussion is that the Actor Model was specifically designed and architected to serve as a concurrent model whereas the object-oriented model is not. More specifically, in a Scala actor system, actors interact and share information, without any presupposition of sequentiality. The mechanism by which actors share information with one another, and ask one another, is message passing. AKKA framework is a toolkit and runtime for building highly concurrent, distributed, and fault-tolerant applications on the JVM [5]. Akka is written in Scala, with language bindings provided for both Scala and Java. Akkas approach to handling concurrency is based on the Actor Model. Akka creates a layer between the actors and the underlying system such that actors simply need to process messages. Upon receiving a message, an actor may take one or more of the following actions:

- Execute some operations itself (such as performing calculations, persisting data, calling an external web service, and so on)
- Forward the message, or a derived message, to another actor
- Instantiate a new actor and forward the message to it

B. *Technology Profile:*

1) *Frameworks:*

- Play Framework: Play Framework makes it easy to build web applications with Java & Scala. Play is based on a lightweight, stateless, web-friendly architecture. Built on Akka, Play provides predictable and minimal resource consumption (CPU, memory, threads) for highly-scalable applications.
- Akka Framework: Akka is a toolkit and runtime for building highly concurrent, distributed, and fault-tolerant applications on the JVM. Akka is written in Scala, with language bindings provided for both Scala and Java [5].

2) *Encoding:*

- UTF-8: UTF-8 (8-bit UCS/Unicode Transformation Format) is a variable-length character encoding for Unicode. It is the preferred encoding for web pages.

3) *Database Support:*

a) PostgreSQL:

PostgreSQL, often simply Postgres, is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance. It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.

4) *Languages:*

a) JAVA:

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers ”write once, run anywhere” (WORA).

b) SCALA:

Scala is an acronym for Scalable Language. Scala is a pure-bred object-oriented language.

V. IMPLEMENTATION

The Application is divided into two parts:

- Front End:
- Back End:

A. *Front End:*

The Front End module focuses on the Flow of the Application from the Merchant side to the Payment gateway. It includes the process of transferring encrypted data from the merchant system to the payment gateway, implementation of different payment modes like debit, credit, and net banking payments and forwarding the data to the bank and storage of the appropriate status from the bank.

Flow of the Application from the Merchant side to the Payment gateway: Data Transfer using HMAC: Details entered by the customer like the client-id, the amount on the Merchant site are then send to the Payment Gateway. Now to send the data free from errors, HMAC is used. First, the data entered by the customer i.e. details are encrypted using a symmetric key which is stored in the database of the Payment gateway and a H-MAC is generated. Now the data plus the H-MAC is sent to the Payment gateway for further processing. In this part, the payment gateway will generate its H-MAC which will be compared with the one received. If they are equal then we can say that the data is free from errors, else the data has been modified.

Implementation of different Payment Modes: After the data has been checked using HMAC, then card details are taken from the customer at the Payment gateway side and the Payment mode is checked, and action is taken accordingly.

1) For Debit Card Payment & For Credit Card Payment:
The card details for this type of Payments are very crucial. These can be tampered. Hence, a DEK key encryption is used to store the card details in the database. The details are first stored in the in-memory database and when a response is returned from the bank, the response plus the card details (in encrypted format) are stored in the database for further settlement.

2) For Net Banking Payment:
The customer is asked forthe Bank name from which the customer wants to proceed the transaction. Then the Payment gateway will accordingly forward the customer to the particular Bank site on which the customer enters its username and password for authorization.

B. Back End:

The Back End module includes the administration information of the transactions taking place. There will be a login portal for Customer, Merchant and Payment gateway admin. After login the Customer will have a view of:

1) The Buy Online Product Page: The customer will select the product he wants to buy and selects the merchant from which he wants to buy the product.
2) Profile: The customer will be able to view his details that he used to create an account.
3) Feedback and Complaints: The customer can log his complaints and give feedback.

The Merchant System Admin will have the following view:

- Account Details: The Merchant view summary like account information, payment methods, merchant details, change password.
- Customer Payments These are categorized as follows:

1) Flagged Transactions: These Transactions are kept on hold by the monitoring system. Such Payments are manually verified and changed the status to authorized or canceled.
2) Authorized Transactions: The customer will be checked for its card details and balance in the account through the payment gateway.
3) Captured Payments: These Transactions are selected by the Merchant System for further processing to capture the amount of the product.
4) Canceled Payments: These Transactions are those which are canceled by the merchant side.
5) Failed Payments: Bank declined transactions will be listed in this category.
6) Incomplete Payments: Due to some reason these payments remain incomplete.
7) Chargeback: A chargeback is a consumer originated recall of funds taken but not authorized beforehand by the merchant. Consumers typically start a chargeback by communicating with their issuing bank or credit card provider.
8) Refund: A refund is frequently defined as a simple return of funds from the merchant to the consumer. An example of a refund: the merchant does not ship the product after receiving payment; the consumer notifies the merchant of their dissatisfaction with the purchase, and the merchant then issues a credit to cancel the charge. The Merchant System will have analysis report of the customers which selected that merchant for processing its transaction request.

1) Payment Gateway Administrator:
The Payment gateway admin will have a complete view of all the merchants transactions and can change or modify their status. It can also have the analysis report of how many transactions were successful, canceled, failed, authorized and captured. It can also add merchants to its system and view the merchants which are using the Payment gateway.

VI. CONCLUSION

Implementation of the proposed payment gateway is based on the Akka framework. By using the actor model of the Akka framework and Scala, we can increase the scalability of the existing payment gateways. It is possible to implement strong security payment protocols on it. Compared to the existing systems, this system costs very less and is more scalable and extensible. Detailed working flow and architecture of the system is outlined. We will evaluate the performance and security in the future.

REFERENCES

[1] Bo Meng, Qianxing Xiong. The 8th International Conference on Computer Supported Cooperative Work in Design Proceedings. [Research on Electronic Payment Model]. ,2004.
[2] Ailya Izhar, Aihab Khan, Malik Sikandar Hayat Khiyal, Wajeeh Javed, Shiraz Baig. Journal of Theoretical and Applied Information Technology [Designing and Implementation of Electronic payment gateway for developing countries]. ,2011.
[3] Kai Wei, Yih-Farn Robin Chen, Alan J. Smith, Binh Vo. Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS06) [WhoPay: A Scalable and Anonymous Payment System for Peer-to-Peer Environments]. ,2006.
[4] Gateway Index, Processing Speed 2015. http://gatewayindex.spreedly.com/
[5] Akka Documentation, Actor Model 2015. http://doc.akka.io/docs/akka/snapshot/general/actors.html