Some Design Considerations for a Mobile Payment Architecture

Praveen Chandrahas, Deepti Kumar, Ramya Karthik,
Timothy Gonsalves, Ashok Jhunjhunwala and Gaurav Raina
Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai 600 036, India
Email:{chandrah,gaurav}@cse.iitm.ac.in, {deepti.kumar,ramyakar}@gmail.com,
tag@lantana.tenet.res.in, ashok@tenet.res.in

***Abstract*—The rapid proliferation of mobile phones now provides an opportunity to harness their potential towards financial transactions. One form of such financial transactions is mobile payments. Our focus, in this paper, will be on certain design considerations for a mobile payment architecture.**

For widespread adoption, interoperability is a key concern. We first outline a previously proposed architecture, which supports interoperability. The design of this architecture requires the user to know only the beneficiary's mobile number in order to initiate a mobile payment. This is restrictive in the sense that only one bank account can be linked to a mobile number. To enhance flexibility it would be desirable, especially for merchants, to be able to link multiple bank accounts to a single phone number.

We propose an alternate and enhanced design that allows the flexibility to link multiple bank accounts while also allowing the transactions to be conducted with just the mobile number.

We evaluate and compare these two designs on various criteria. The details of implementation issues, advantages and limitations are presented. The analysis is a step towards the evaluation process of various design choices for mobile payment architectures.

I. Introduction

A mobile payment can be defined as the *transfer of money from one entity to another entity through the exchange of information via a mobile device*. Mobile devices may include mobile phones, PDAs, wireless tablets and any other device that may connect to mobile telecommunication network and make it possible for payments to be made. The mobile phone helps in the exchange of information required for the actual transfer of money [3] [4].

For any mobile payment architecture to be widely adopted, the following challenges need to be overcome: (1) Interoperability, (2) Usability, (3) Simplicity, (4) Universality, (5) Security, (6) Privacy, (7) Trust, (8) Cost, (9) Speed and (10) Cross-border payments [4]. The regulatory environment that we focus on is where every mobile payment transaction goes through the bank accounts of the two parties involved in the transaction; in essence, the flow of the transaction can be described simply as *customer-bank-bank-customer*. The mobile phone acts only as an instrument to access the bank account.

It is natural to expect that customers can have very different telecommunication service providers, and will be registered with different banks. Thus interoperability, which is defined as the mechanism which allows a mobile payment to be made irrespective of the bank and/or the service provider to which the customers belong to, becomes an integral aspect in ensuring the widespread adoption of mobile payments.

An architecture was proposed previously in [1] [2], which addresses the concern of interoperability. A key limitation of the design of the architecture is that it is not flexible enough to accommodate multiple bank accounts to be mapped to a single mobile number. The main objective of our work is to consider an alternate design for the architecture which addresses this issue. We also enumerate the challenges faced by our proposed design choices and then proceed to give solutions to those challenges.

The rest of the paper is outlined as follows. In section II, we give a brief background of the previously proposed architecture and highlight its advantages and disadvantages. In section III, we propose an alternate design for the architecture which takes into consideration some of the issues we raised. In section IV, we conclude with our key contributions.

II. A Previously Proposed Architecture

We now recapitulate the architecture proposed in [1]; also see [2] for further reference. This architecture allows the mapping of a mobile number to a *single* account number. It enables a customer to initiate a mobile payment just by knowing only the mobile number of the other person. A skeletal structure of the architecture is represented in Figure 1.

The architecture introduces the term Mobile Payment Provider (MPP). An MPP can be a third-party who provides an interface between the end-user and the Bank. The person initiating the transaction is called the *customer* and the person who receives it is called *beneficiary*. A customer initiates a mobile payment by entering the mobile number of the beneficiary. This information is sent across from the Telecommunication Service Provider (TSP) to the MPP which in-turn communicates the same to the customer's bank. After the appropriate processing at the customer's bank, the transaction is sent to the beneficiary's bank. An important step here is to identify the beneficiary's bank. This can be accomplished by storing all the mapping data in a *Central Repository*.

Every bank is assigned a unique identification number, referred to as the *bank id*. The information that has been passed

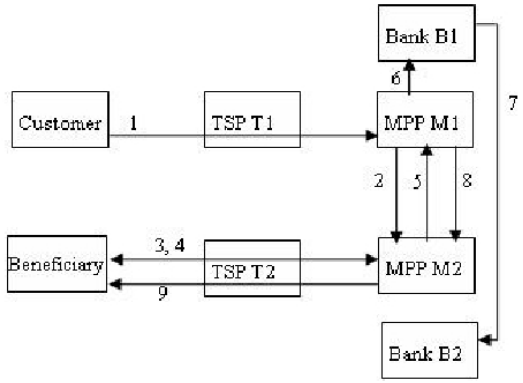

Fig. 1. Previously Proposed Architecture

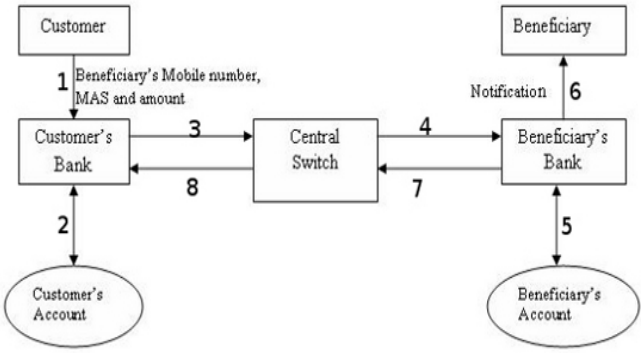

Fig. 2. Proposed Design Changes

to the central repository contains the beneficiary's mobile number. A search is performed on the central repository to find the bank id of the beneficiary after the customer's bank authenticates the transaction. The search key is the mobile number of the beneficiary. Once the beneficiary's bank id is found, the transaction is then routed to the beneficiary's bank where further processing of the transaction takes place.

A. Evaluation

This architecture is the simplest to use from the end-user's point of view. This is because the only thing the customer has to know is the mobile number of the beneficiary.

The cost of a transaction depends the additional network infrastructure required by the architecture. The architecture requires that a central repository be built for the purpose of routing. And banks should bear the cost of the MPP. Other than these factors, the architecture makes use of the existing telecommunication network infrastructure.

B. Limitations

One key limitation is that it restricts a mobile number from being mapped to multiple bank accounts. The design maximises usability while restricting flexibility. The design is also insensitive to mistakes made by the customers while entering the mobile number of the beneficiary.

III. An Alternate Design

In the light of the above discussion on the limitations of the previously proposed design, we propose a new design which addresses the concerns raised in the previous section. The main aim of the proposed design is to allow multiple accounts to be linked to a single mobile number and to provide for a safety-net which can act as a safeguard when a customer enters an incorrect mobile number.

A. Multiple Accounts

The additional feature of multiple accounts comes at the cost of restricting simplicity for the end-user. In order to allow for multiple accounts, we introduce the term *'Mobile Account Selector'* (MAS). When a customer registers with a bank for mobile payments, the bank assigns a 3-digit number. This number, which we call the MAS, uniquely identifies an account across any bank when used in combination with the mobile number. Every account which is linked to a mobile number can be uniquely identified by <mobile_number, MAS> combination. In order to satisfy this condition of uniqueness, we have to impose certain restrictions on MAS which are discussed in Section III-D.

A mobile payment is initiated by entering the beneficiary's mobile number, beneficiary's MAS and the amount of money to be transferred. This information is first sent to the customer's Bank/MPP. After authentication, the transaction is sent to the central repository, which we call the *central switch*. The switch has a collection of the records pertaining to <mobile_number, MAS, bank_id>. The switch does a search in its database for the <mobile_number, MAS> combination. On a successful search, the bank id is found and the transaction is routed to the beneficiary's bank where further processing of the transaction takes place.

Every bank needs to maintain a database mapping the <mobile_number, MAS> combination with the appropriate account number. This database corresponds only to the accounts of that particular bank. When the information is sent across from the central switch, the beneficiary bank needs to do a look-up on this database to find the actual account number to which the credit should take place. The proposed architecture is captured in Figure 2.

The basic objective of the design is to maximise flexibility. In order to maximise both usability and flexibility, we introduce the idea of a *default account*.

B. Default Account

In order not to affect the usability when a customer prefers not to have multiple accounts, a customer can enable a default account. When a customer registers for mobile payment with a bank, they can choose to specify a particular account as the default account. This allows transactions to be done *without* entering the MAS as well. When a transaction is initiated by entering only the mobile number of the beneficiary, the central switch checks to see if a default account has been registered with the mobile number. If a default account is registered, then the switch routes the transaction to the appropriate bank. Thus,

the usability of the architecture is not affected significantly when a subscriber choses to have a default account linked to their mobile. A customer can choose to select a default even if multiple accounts are linked to their mobile number. The MAS can then be used only when money has to be transferred to an account other than the default account. A slight change in the implementation of the central switch is required in order to implement this.

C. Issues

The proposed design throws up *two* major implementation challenges. The first of these challenges is the generation of MAS. The combination of <mobile_number, MAS> should uniquely identify an account number across the banks. MAS also serves other purposes which are enumerated in Section III-D. The second challenge is the storage mechanism for building the central switch. The central switch should facilitate extremely fast look-ups while facing certain constraints. These issues are addressed in the subsequent discussion.

D. MAS Generation

Requirements

In order to satisfy the goals, it is imperative that we put some constraints on the generated number. These include, but are not limited to the following:

Mandatory Constraints

- The MAS should uniquely identify an account number when used in conjunction with a Mobile Number.
- A large percentage of customers should not be holding the same MAS. This follows from the goal of the MAS to act as a second safety net. In case a large percentage of the customers do hold the same MAS, the probability of an incorrectly entered mobile number leading to a fund transfer to an unintended party will increase significantly.

E. Some Options for Generation of MAS

1. Random Number Generator (RNG)

(a) Description:

This method makes use of a random number generator for generating the MAS. The steps in the algorithm given below are iterated until a suitable MAS is generated.

(b) Algorithm

1) MPP/Bank generates a 3-digit random number, **R**, in the range [000,999] using an RNG.
2) Send the generated number along with the Mobile Number of the customer and the current account number to the central switch.
3) Central switch checks the count of the number of accounts already mapped to the mobile number. If the count equals the value of 1000, then the request is rejected specifying the same reason.
4) Else, central switch checks for the number **R** in the list of MAS's associated with the current mobile number.
5) If number **R** is already present, inform the MPP/Bank of the same and go to step (a).
6) Else, designate the number **R** as the MAS of the current account and intimate the same to the MPP/Bank.

(c) Constraint Satisfaction

Both the constraints are guaranteed.

(d) Efficiency

The efficiency will come down when trying to generate MAS for a mobile number which already has a large number of accounts linked to it.

2. Incremental Approach

(a) Description

The incremental approach eliminates the drawback of RNG method. This approach incorporates elements of randomness and determinism to generate MAS. The method is best illustrated by the Algorithm 1 in Section-III.

(b) Algorithm

Algorithm 1 Generating MAS using the Incremental Approach

```
1: Customer approaches MPP/Bank for mobile payments
2: MPP/Bank sends a request to Central switch; Request contains Mobile Number and Account Number of the customer
3: Central switch:
4: if Customer already registered for mobile payments then
5:     if count < 1000 then
6:         if T = 0 then
7:             if L+1 ≠ F then
8:                 L ← (L + 1)
9:                 increment counter
10:                return L as MAS
11:            else
12:                T ← 1 {Counter is about to be tipped after this operation}
13:                Search for free slots
14:                Increment Counter
15:                return First Free Slot as MAS
16:            end if
17:        else
18:            Search for free slots
19:            Increment Counter
20:            return First Free Slot as MAS
21:        end if
22:    else
23:        Maximum number of accounts reached;
           Request rejected
24:    end if
25: else
26:    F ← Random 3-digit number
27:    Increment Counter
28:    L ← F
29: end if
```

(c) Constraint Satisfaction

Satisfies both.

(d) Efficiency

The efficiency is not affected by the number of accounts already linked. The general performance is very good.

F. Storage Mechanisms

The architecture requires that the transaction be routed to the appropriate bank based on the <mobile_number, MAS> combination received by the central switch. The main bottle-neck for performance arises during the look-up process. As the number of subscribers keep increasing, the need for a fast, reliable and robust look-up system is much more ominous. We present some methods and highlight the advantages and disadvantages, cost and performance issues associated with the methods that we propose.

1. Database Based

We chose MySQL as it offers the flexibility to let the storage engine be selected depending on the specifications. The simulation that was run to collect statistics consisted of 200 million records. The tests were conducted on a system running on Intel Core-2 processor with 4-GB RAM and Linux based operating system. The following set of queries were run to obtain the search times:

- Query for a random mobile number, which *exists* in the database, without specifying the MAS.
- Query for a random mobile number, which *exists* in the database, along with the MAS.
- Query for a random mobile number, which *does not exist* in the database, without specifying the MAS.
- Query for a random mobile number, which *does not exist* in the database, along with the MAS.

MySQL Storage Engines

(a) MyISAM

MyISAM has built-in full-text search which allows for faster search. Building indexes on the appropriate columns leads to an increase in performance, though it is a slight trade-off with memory requirements. In fact, as the results in Table I indicate, building an index results in a substantial performance gain over a non-indexed table.

A significant difference in query times can also be found when a look-up is performed based on the <mobile_number, MAS> combination rather than just the mobile number. Building an index over these two columns, though resulting in increased disk usage, will lower the search times by an order of magnitude.

(b) MEMORY (previously *Heap Engine*)

The MEMORY storage engine is designed to be an in-memory storage for extremely fast access and low latency. MEMORY engine allows extremely fast look-ups by creating a hash of the data. With the data being static, there are better chances of coming up with a perfect hash function which allows constant order look-up of data. Creating an index will not affect the performance of MEMORY engine as much as the MyISAM engine, as evidenced in Table II.

(c) Clustered Database

Both (a) MyISAM and (b) MEMORY above, fare well in terms of performance. However, in order to meet the criteria for an optimal central switch, reliability and scalability are also essential. This can be achieved by using a Clustered Database. Different storage nodes can be configured to run as a single cluster. The cluster architecture is a *shared-nothing architecture* which means that there is no single point of failure. The data nodes can be configured to be replicas of each other ensuring high reliability. Scalability follows from the same principles which allows reliability. Adding additional data nodes leads to increased capacity. Performance can be maintained at the same level even after the increased disk-usage by adding in more SQL servers. The load can be balanced among the SQL servers which in-turn, distribute the load among the data nodes. In the light of the above discussion, we summarise the advantages and disadvantages of databases as follows:

Advantages

- Database based storage solutions are cheaper compared to hardware solutions like Content Addressable Memory (CAM). The trade-off is between performance and cost.
- Reliability and Scalability are easy to achieve when using a database based storage mechanism.

Disadvantages

- Databases are not optimised to store *one-to-many* relationships. This can adversely impact the performance when there are a large number of one-to-many relations. This is the typical scenario in the proposed architecture for mobile payments. When a mobile number is linked to x number of accounts, the database will have x unique records for the same mobile number.
  One alternative would be to use a hash table. The mobile number can be hashed using an appropriate hash function. To minimise the effect of collisions, and to enable multiple accounts to be searched in a cost-effective manner, separate chaining [7] could be used. Instead of the traditional list used in separate chaining, we propose an alternate data structure based on a tree.

2. Tree Based

The main aim here is to avoid comparing the same mobile number multiple times once it is known that the given pattern does not match the mobile number. The Abstract Data Type (ADT) for the tree, in pseudo 'C', is shown in Algorithm 2.

Though the ADT is based on a Binary Search Tree, any self-balancing tree, such as Red-Black trees [8] or AVL Trees [9], can be used. The MAS associated with a mobile number is stored in a double-dimensional array which also holds the corresponding bank id. The search for a <mobile_number, MAS> combination will first lead to a binary search, *findNode*, for the mobile number. Once the mobile number is found, the node containing the mobile number is passed to another function along with the MAS from the original search pattern. This function, which we call *findBin*,

	Mobile No.	Mobile No. + MAS	Non-Existent Mobile No.	Non-Existent Mobile No. + MAS
With Index	$0.04s$	$0.05s$	$0.02s$	$0.03s$
Without Index	$2.35s$	$3.27s$	$2.54s$	$2.68s$

TABLE I
QUERY TIMES FOR MYISAM STORAGE ENGINE

	Mobile No.	Mobile No. + MAS	Non-Existent Mobile No.	Non-Existent Mobile No. + MAS
With Index	$0.03s$	$0.06s$	$0.03s$	$0.04s$
Without Index	$1.15s$	$1.27s$	$1.54s$	$1.68s$

TABLE II
QUERY TIMES FOR MEMORY STORAGE ENGINE

will search the double-dimensional array for a match with the MAS from the original search pattern. Once a match is found, the corresponding entry for the bank id is returned.

Algorithm 2 Tree Abstract Data Type

$struct$ node{
$struct$ node$*$ left;
$struct$ node$*$ right;
int mobile_number;
int $**$mas;}

search Function

search(mobile_number,int mas)
$struct\ node*\ x \leftarrow$ **findNode**(int mobile_number);
return findBin(x ,mas);

findNode Function

findNode(int)
return $struct\ node*\ z$ such that $z \rightarrow$ mobile_number $= x$

findBin Function

findBin($struct\ node*$, mas)
return $z.mas$ where $z.mas = mas$

3. Tree Based on Data Nodes

If the tree-based data structures are used without a hash table, the aforementioned Tree Based method can be further refined by grouping together, in bunches, the mobile numbers. A group of mobile numbers can be formed after taking into account the way the mobile numbers are distributed. In a typical scenario where there are a large number of mobile users, grouping the mobiles based on the first 4 digits of the mobile number might serve the purpose. We call these groups ***bins***. Based on the number of bins formed, these bins can now be arranged in a tree structure. If the number of bins is smaller than a certain threshold value, then they can be arranged in an *n-ary* tree. The threshold value depends on the number of bins. Varying the number of mobiles per Bin and the number of Bins, different levels of performance can be obtained.

IV. CONCLUSIONS

In the design of architectures for mobile payments, there is a potential tussle between usability and flexibility. In this paper, we outlined a design option that allows for the flexibility to have multiple bank accounts linked to a single mobile phone number.

The proposed architecture complies with the major issue of interoperability; however, it puts a major fraction of the load on a central switch. The details of the implementation issues, the advantages and the limitations were analysed from various perspectives.

When real pilots are conducted, and we get real implementation data, then that may further bring to light the design trade-offs in mobile payment architectures. Currently, a pilot is under way in India with several banks.

ACKNOWLEDGMENTS

The authors would like to acknowledge funding by the Department of Information Technology, Government of India for the project entitled 'Mobile Payment Certification Lab'. Additionally, the Technology Committee members of the Mobile Payment Forum of India are acknowledged for their role in the development of the Interoperability Standards for Mobile Payments.

REFERENCES

[1] Deepti Kumar, Timothy Gonsalves, Ashok Jhunjhunwala and Gaurav Raina "Mobile payment architectures for India," *National Conference On Communications*, 2010.
[2] Deepti Kumar "Mobile Payments: Interoperability and Implementation," M. S. thesis, Indian Institute of Technology Madras, India, 2009.
[3] Y. A. Au and R. J. Kauffman "The economics of mobile payments: Understanding stakeholder issues for an emerging financial technology application," *Electronic Commerce Research and Applications*, 2007.
[4] S. Karnouskos and F. Fokus "Mobile Payment: a journey through existing procedures and standardization initiatives," *IEEE Communications Surveys and Tutorials*, pp. 44–66, 2004.
[5] Cellular Operators Association of India, *Subscriber Figures for 2009*. Available at: `http://www.coai.com/statistics.php`
[6] *http://www.economist.com/node/14505519*
[7] Paul E. Black, "Separate Chaining," *U.S. National Institute of Standards and Technology*, 2010. [Online]. Available at: `http://xw2k.nist.gov/dads/HTML/separateChaining.html`. [Accessed Nov.3, 2010]
[8] Leonidas J Guibas and Robert Sedgewick "A Dichromatic Framework for Balanced Trees," *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, 1978.
[9] Adelson-Velskii, G. and E. M. Landis "An algorithm for the organization of information," *Proceedings of the USSR Academy of Sciences*, vol. 146, pp. 263-266, 1962.