SMS Based Payment System

Author - Vipul Mehta

Summary

This document defines an SMS based payment system which can facilitate two entities to execute a monetary transaction securely via SMS exchange with a server. It provides an overview of the implementation and the flow of money transaction.

Tools Used

Node.js and Express.js – server side development.

Azure for server hosting.

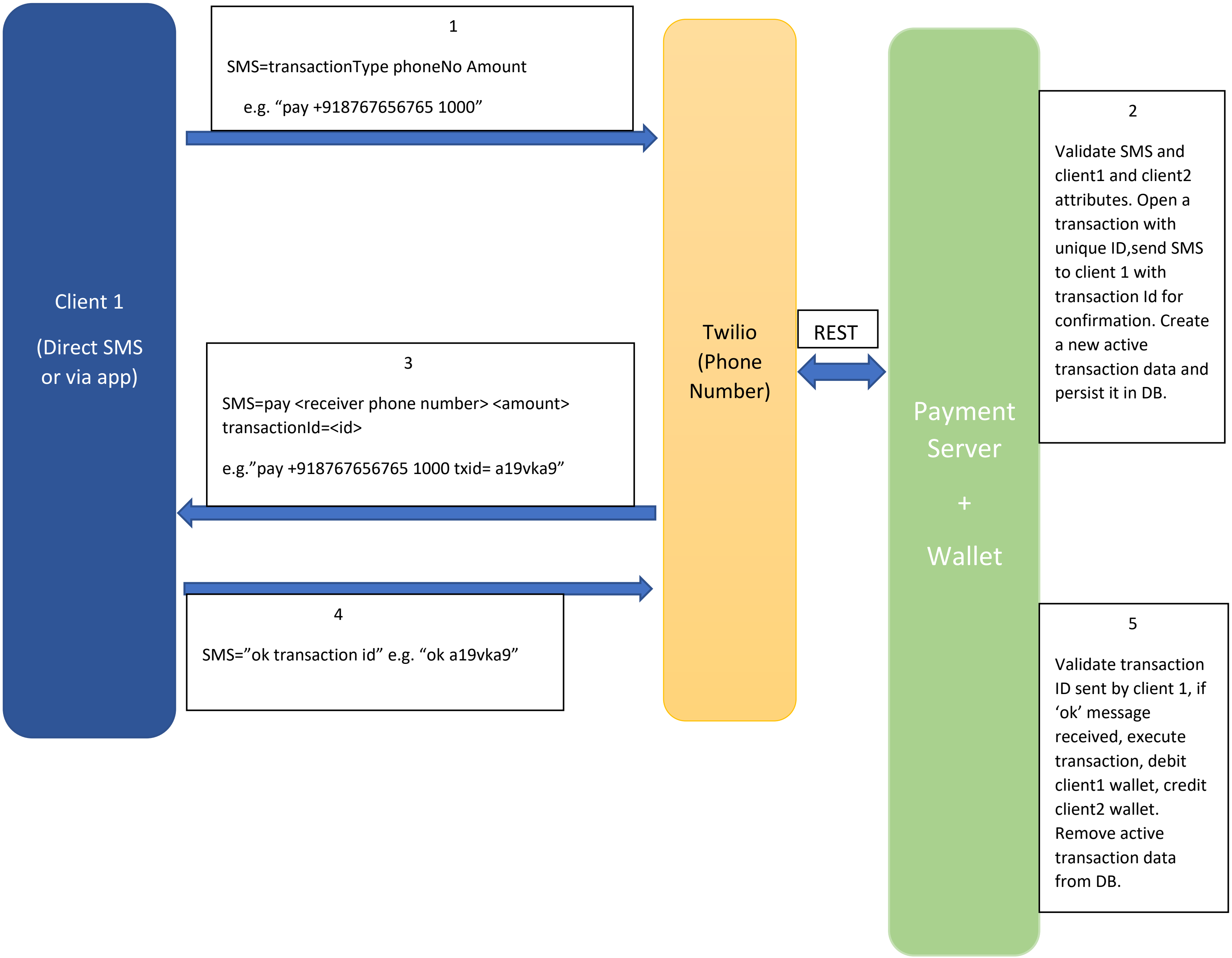
MongoDB - server side data storage.

Twilio - communication API for sending and receiving SMS on server side

Android – Client side app

Transaction Flow

1) Client 1 paying to Client 2



Client 1

(Direct SMS or via app)

Twilio (Phone Number)

REST

Payment Server

+

Wallet

1

SMS=transactionType phoneNo Amount

e.g. “pay +918767656765 1000”

2

Validate SMS and client1 and client2 attributes. Open a transaction with unique ID,send SMS to client 1 with transaction Id for confirmation. Create a new active transaction data and persist it in DB.

3

SMS=pay <receiver phone number> <amount> transactionId=<id>

e.g.”pay +918767656765 1000 txid= a19vka9”

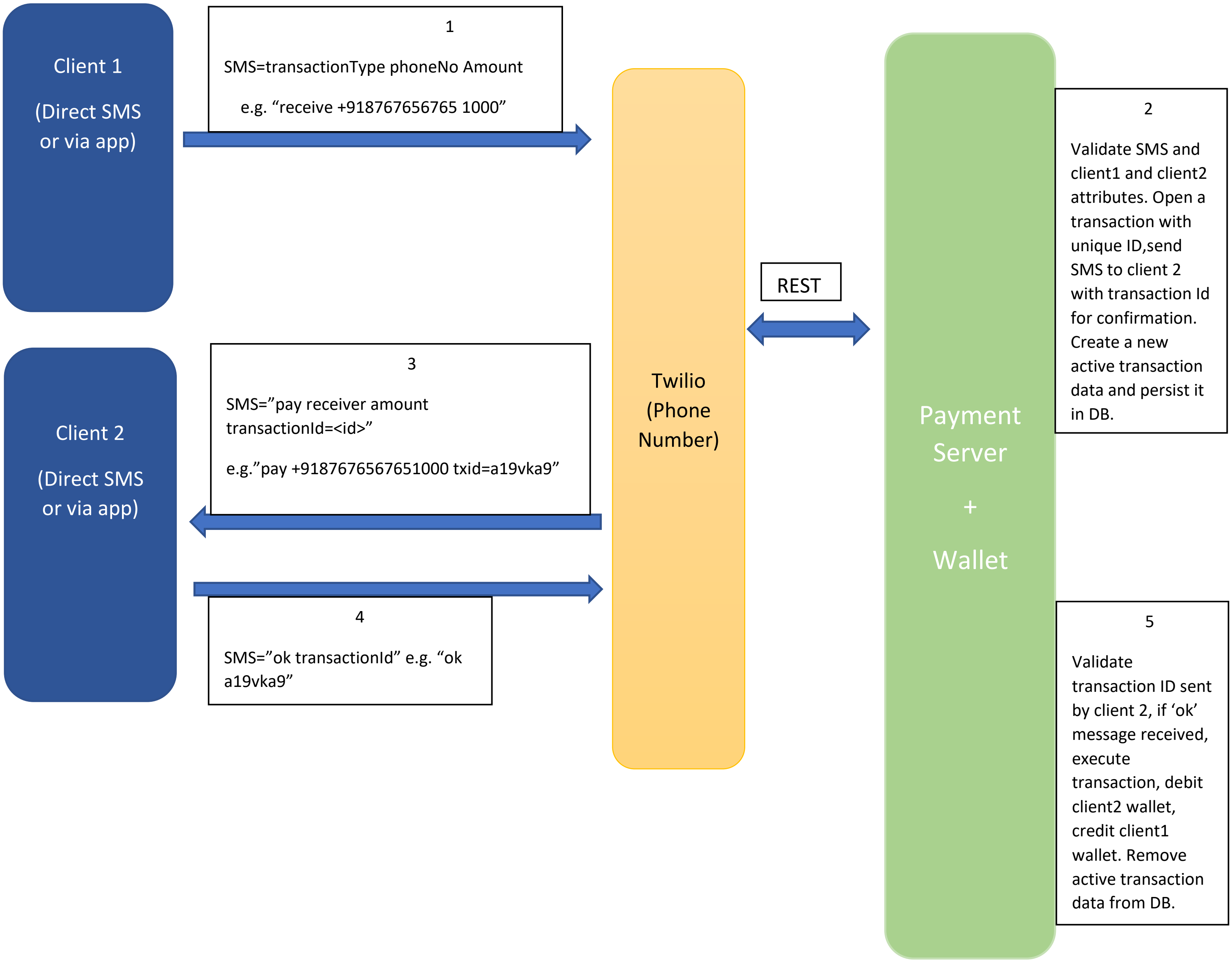
4

SMS=”ok transaction id” e.g. “ok a19vka9”

5

Validate transaction ID sent by client 1, if ‘ok’ message received, execute transaction, debit client1 wallet, credit client2 wallet. Remove active transaction data from DB.

2) Client 1 requesting a payment from Client 2



Client 1

(Direct SMS or via app)

1

SMS=transactionType phoneNo Amount

e.g. “receive +918767656765 1000”

Client 2

(Direct SMS or via app)

3

SMS=”pay receiver amount transactionId=<id>”

e.g.”pay +9187676567651000 txid=a19vka9”

4

SMS=”ok transactionId” e.g. “ok a19vka9”

Twilio (Phone Number)

REST

Payment Server

+

Wallet

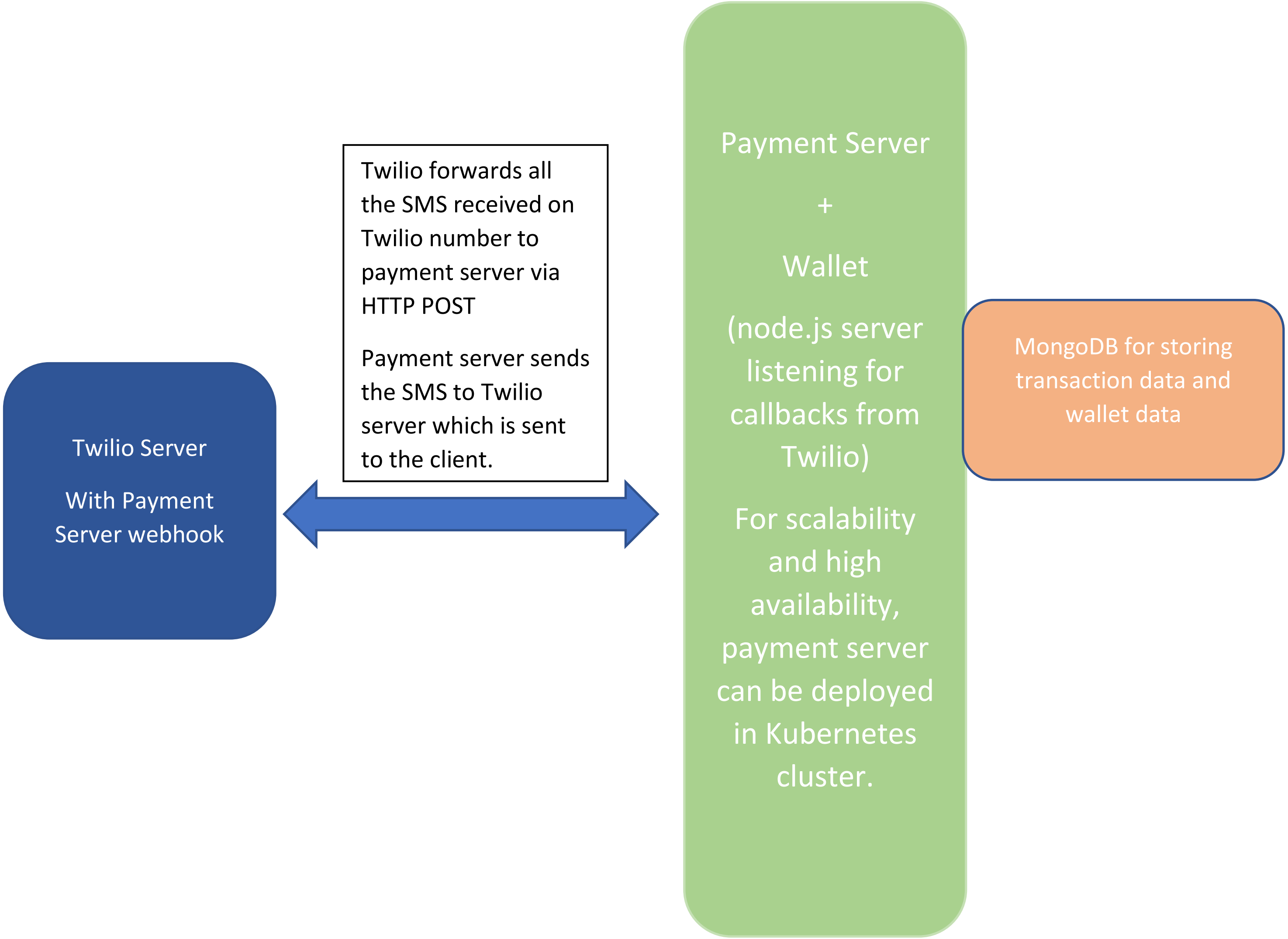
2

Validate SMS and client1 and client2 attributes. Open a transaction with unique ID,send SMS to client 2 with transaction Id for confirmation. Create a new active transaction data and persist it in DB.

5

Validate transaction ID sent by client 2, if ‘ok’ message received, execute transaction, debit client2 wallet, credit client1 wallet. Remove active transaction data from DB.

Server Architecture



Twilio Server

With Payment Server webhook

Twilio forwards all the SMS received on Twilio number to payment server via HTTP POST

Payment server sends the SMS to Twilio server which is sent to the client.

Payment Server

+

Wallet

(node.js server listening for callbacks from Twilio)

For scalability and high availability, payment server can be deployed in Kubernetes cluster.

MongoDB for storing transaction data and wallet data

Transaction Security

Twilio REST calls to payment server are over HTTPS.

SMS exchange between client and payment server can be further secured by implementing PKI (public key infrastructure). Client and payment server can do initial public key exchange and create a shared secret key for encrypting the SMS.

Private Key of Client = c

Public Key of Client = cG

Private Key of Server = s

Public Key of Server = sG

After public key exchange:

Client secret key = c X sG = csG

Server secret key = s X cG = csG

To prevent man in the middle attack, server needs to use another private key and public key certificate.

Server should sign the transaction message with certificate private key and send it along with the public certificate serial number in the SMS (multi-part).

Client should verify the signature using server public certificate packaged into the App. Any attacker will not be able to create the required signature + client will not trust its certificate.

Feasibility and Scope

- The proposed system depends on SMS only, so it works even when the two peers are offline.
- For smartphones, it is easy to develop app with inbuild PKI support. For feature phones it depends on availability of phone SDK for development. In worst case, just normal SMS will also work (but less secure) but end user needs to know the transaction SMS format.