Question 3 Report

Kshitij Nigam
2023202031

1. Broad Overview of the System

The system built here is a miniature version of Stripe, a payment gateway that interfaces with various bank servers to manage transactions. The payment gateway facilitates secure communication between clients and bank servers, processes payments, handles authentication, authorization, and logs events for transparency and troubleshooting. It supports essential features like registration, authentication, balance retrieval, payment processing, idempotent payments, and offline payments. The system uses gRPC for efficient communication between clients, the payment gateway, and bank servers, with support for SSL/TLS encryption for secure data transfer.

Key components of the system include:

- **Payment Gateway**: The central entity responsible for processing payments, handling client registrations, authenticating users, and coordinating transactions between clients and bank servers.
- **Bank Servers**: Represent individual banks and handle the preparation, commitment, and abortion of transactions as part of the two-phase commit protocol (2PC).
- **Clients**: End-users who interact with the payment gateway, register, authenticate, check balances, and make payments.

The system also implements features like offline payments (queued for retry), idempotent payments (to prevent duplicate transactions), and logging for system health monitoring and debugging.

2. Design Choices for Each Requirement

a) Secure Authentication, Authorization, and Logging

- **Authentication**: The system ensures only authorized users can access the payment gateway by requiring clients to authenticate via username and password. Credentials are stored securely in a database, and communication between the client and server is secured with SSL/TLS. This is achieved using gRPC's authentication support.
- **Authorization**: The authorization mechanism ensures that only authenticated users can access sensitive features like checking balances and initiating payments. Authorization checks are done using gRPC interceptors, which inspect incoming requests and validate the client's token.
- **Logging**: Logging is essential for tracking system behavior and errors. The system uses logging interceptors on both the client and server sides to log every request, response, and error along with important metadata such as client IDs and transaction details.

b) Idempotent Payments

- The system handles idempotency by generating a unique idempotency key for each payment. If a client retries the same transaction, it will use the same key, preventing multiple deductions. The payment gateway stores the status of each transaction in a database, ensuring that repeated requests for the same transaction are handled gracefully.

c) Offline Payments

- The system queues payments if the client is offline. When the client regains connectivity, the system retries the queued payments. This is implemented using a thread-safe queue structure, where each payment request is stored with an idempotency key. Only unprocessed payments are retried.

d) Two-Phase Commit (2PC) with Timeout

- The system implements a two-phase commit (2PC) protocol for processing payments between banks. The payment gateway acts as the coordinator, while the sending and receiving banks are the participants. Each bank prepares for the transaction in phase one, and if both banks are ready, the transaction is committed in phase two. If any bank fails or if a timeout occurs, the transaction is aborted. This ensures atomicity and consistency across multiple banks.

3. Assumptions and Future Work

a) Assumptions

- The system assumes that bank servers and payment gateway servers are running on the same network and are reachable.
- The implementation uses SQLite as a database for simplicity and assumes that it is suitable for the scope of the assignment.
- Authentication is done using basic username/password validation, and the system does not handle complex multi-factor authentication (MFA) or session management.
- In a production system, the database and transaction systems would be more robust, scalable, and fault-tolerant, but for this assignment, SQLite is sufficient.

b) Future Work

- **Scalability**: The system can be enhanced to handle multiple clients and concurrent transactions more efficiently using a distributed database and queue management system.
- **Multi-factor Authentication (MFA)**: Adding additional layers of security such as OTPs or biometric authentication can further strengthen the system.
- **Transaction Rollback**: While the system currently supports transaction abortion in case of failure, more complex rollback mechanisms could be added to support scenarios such as partial payments or retries across multiple systems.
- **Database Optimization**: SQLite is used for simplicity, but for a real-world system, using a distributed database like PostgreSQL or MySQL would improve scalability and reliability.

4. Other Key Points

- **Client-Server Communication**: The system uses gRPC, which is a modern, high-performance RPC framework that ensures efficient communication between clients, the payment gateway, and bank servers. gRPC supports bidirectional streaming, which allows for real-time updates and responses.
- **Security Considerations**: SSL/TLS encryption ensures that communication between the client and the payment gateway is secure, preventing man-in-the-middle attacks. Additionally, sensitive information like passwords is transmitted in an encrypted format.
- **Database Integrity**: The system maintains transactional integrity using SQL transactions, ensuring that operations such as balance updates and payment processing are atomic and consistent.