Ira A. Fulton Schools of Engineering
Arizona State University

CSE 531: Distributed and Multiprocessor Operating Systems

Logical Clock Written Report

Problem Statement

The problem statement asks us to implement Lamport's logical clock algorithm in a distributed banking system consisting of customer and branch processes. This system handles customer requests for deposit and withdrawal, and branches propagate updates. The goal is to ensure that events are correctly ordered based on their logical timestamps, establishing a happens-before relationship between events.

Goal

The goal is to achieve proper coordination and ordering of events in a distributed banking system by implementing Lamport's logical clock algorithm in which inter-process communication is established via gRPC. This is accomplished by the following objectives:

- Define a service in a .proto file.
- Generate server and client code / stubs using the protocol buffer compiler present in the grpcio-tools package in python
- Use the Python gRPC API to create client and server components.
- Implement communication between customer and branch processes and branch and branch processes using gRPC.
- Implement interfaces (Query, Deposit, and Withdraw) for customer-branch and branch-branch interactions.
- Ensure that the happen-before relationship is followed for events by implementing Lamport’s logical clock algorithm.
- Write the events corresponding to the customer and branch processes to the respective output files as mentioned in the sample output file.

This ensures that events, such as customer requests and branch updates, are correctly ordered based on logical timestamps, allowing for consistent and coordinated execution.

Setup

Installations:

- Install Python 3.9 or above (I used Python 3.9).
- Install pip version 9.0.1 or higher.
- (OPTIONAL) If necessary, upgrade your version of pip:
  $ python -m pip install --upgrade pip
  If you cannot upgrade pip due to a system-owned installation, you can run the example in a virtualenv:
  $ python -m pip install virtualenv
  $ virtualenv venv
  $ source venv/bin/activate
  $ python -m pip install --upgrade pip
- Install gRPC:
  $ python -m pip install grpcio
- Install gRPC tools:
  $ python -m pip install grpcio-tools

Running the project:

- Unzip from canvas
- Navigate to the folder where main.py is present.
- This step is OPTIONAL:
  If you want to regenerate the code generated by the gRPC library run:
  $ python -m grpc_tools.protoc -I./protos --python_out=. --pyi_out=. --grpc_python_out=. ./protos/banking_system.proto
  This will regenerate the files: banking_system_pb2.py, banking_system_pb2.pyi, banking_system_pb2_grpc.py
- One main requirement is that if your input file has branches from 1 to n then there should be n empty ports starting from 50001 to run the gRPC branch servers. E.g. If the branch id is 9 then its server runs on port 50009. If the ports are not available on your machine then you can change the value of BASE_PORT in constants.py to a value that suits your machine.
- Run:
  $ python main.py --input_file "./inputs/input_10.json"
  where --input_file is followed by the path of your input file.
- The output will be present in ./outputs/output.json (The path is relative to the folder containing main.py)
- The **3** output files generated are **customer_events.json, branch_events.json, and combined_events.json.**
- The checker files and their corresponding input files are as follows: **checker_part_1.py -> customer_events.json**, **checker_part_2.py -> branch_events.json**, **checker_part_3.py -> combined_events.json**.

Implementation Processes

The implementation processes involve:

- Defining a service in a .proto file for gRPC along with the message format for request and response.
- Generating server and client code / stubs from the .proto file using grpcio-tools package in python
- Creating the branch.py file containing the following:
    - Query functionality
    - Deposit functionality
    - Withdraw functionality
    - propagate_interface functionality using gRPC
    - For the propagate_interface functionality it is important to call other branch stubs with the propagate flag set to False otherwise the program will be stuck in an infinite loop.
    - message_delivery functionality which reads the customer request and performs the action accordingly.
    - The message_delivery and propagate_interface functionality make use of Lamport's logical clock algorithm.
    - Write_branch_message_to_file functionality to write the events from the branch to a file.
- Creating the customer.py file containing the following:
    - execute_events functionality: This reads the events stored in the customers event list and calls the stub for the branch associated with the customer using gRPC. It then stores the response in the customer's recd_msgs. The logical clock of the customer is incremented according to Lamport's logical clock algorithm.
    - get_customer_results: This method returns the results received by the customer in the format mentioned in the output in the project document.
- Creating the main.py file which does the following:
    - create the empty output files
    - read data from the input file
    - start branch processes and initialize their stubs for communicating with other branches and other required data like id, balance and ids of other branches
    - start cutomer processes and initialize their stubs for communicating with their corresponding branch and also initialize other required details like their id and events they have to process.
    - Once all the customer processes are finished stop the branch servers so that the program exits gracefully.

    - Generate the combined output file from the customer_events.json and branch_events.json
- Creating the constant.py file which contains all constants used in the project.
- Creating the logging_util.py file for well formatted customized logging for different processes.
- All interprocess communication has been done using gRPC and it has been ensured that the happens-before relationship is correctly enforced between events of the same process and between send and receive events of the same request.

Results

The customer_events.json files is as follows (Only 1 customer is shown for brevity):


```
[
 {
   "id": 1,
   "type": "customer",
   "events": [
     {
       "customer-request-id": 1,
       "logical_clock": 1,
       "interface": "deposit",
       "comment": "event_sent from customer 1"
     },
     {
       "customer-request-id": 2,
       "logical_clock": 2,
       "interface": "withdraw",
       "comment": "event_sent from customer 1"
     }
   ]
 },
```


The branch_events.json file is as follows (Only 1 branch is shown for brevity):


```
{

 "id": 1,

 "type": "branch",

 "events": [

   {

     "customer-request-id": 1,

     "logical_clock": 2,

     "interface": "deposit",

     "comment": "event_recv from customer 1"

   },

   {

     "customer-request-id": 1,

     "logical_clock": 3,
```

```
    "interface": "propagate_deposit",
    "comment": "event_sent to branch 2"
  },
  {
    "customer-request-id": 1,
    "logical_clock": 4,
    "interface": "propagate_deposit",
    "comment": "event_sent to branch 3"
  },
  {
    "customer-request-id": 1,
    "logical_clock": 5,
    "interface": "propagate_deposit",
    "comment": "event_sent to branch 4"
  },
  {
    "customer-request-id": 1,
    "logical_clock": 6,
    "interface": "propagate_deposit",
    "comment": "event_sent to branch 5"
  },
  {
    "customer-request-id": 1,
    "logical_clock": 7,
    "interface": "propagate_deposit",
    "comment": "event_sent to branch 6"
  },
```

```
{
  "customer-request-id": 1,
  "logical_clock": 8,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 7"
},
{
  "customer-request-id": 1,
  "logical_clock": 9,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 8"
},
{
  "customer-request-id": 1,
  "logical_clock": 10,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 9"
},
{
  "customer-request-id": 1,
  "logical_clock": 11,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 10"
},
{
  "customer-request-id": 2,
  "logical_clock": 12,
```

```
    "interface": "withdraw",
    "comment": "event_recv from customer 1"
  },
  {
    "customer-request-id": 2,
    "logical_clock": 13,
    "interface": "propagate_withdraw",
    "comment": "event_sent to branch 2"
  },
  {
    "customer-request-id": 2,
    "logical_clock": 14,
    "interface": "propagate_withdraw",
    "comment": "event_sent to branch 3"
  },
  {
    "customer-request-id": 2,
    "logical_clock": 15,
    "interface": "propagate_withdraw",
    "comment": "event_sent to branch 4"
  },
  {
    "customer-request-id": 2,
    "logical_clock": 16,
    "interface": "propagate_withdraw",
    "comment": "event_sent to branch 5"
  },
```

```
{
  "customer-request-id": 2,
  "logical_clock": 17,
  "interface": "propagate_withdraw",
  "comment": "event_sent to branch 6"
},
{
  "customer-request-id": 2,
  "logical_clock": 18,
  "interface": "propagate_withdraw",
  "comment": "event_sent to branch 7"
},
{
  "customer-request-id": 2,
  "logical_clock": 19,
  "interface": "propagate_withdraw",
  "comment": "event_sent to branch 8"
},
{
  "customer-request-id": 2,
  "logical_clock": 20,
  "interface": "propagate_withdraw",
  "comment": "event_sent to branch 9"
},
{
  "customer-request-id": 2,
  "logical_clock": 21,
```

```
    "interface": "propagate_withdraw",
    "comment": "event_sent to branch 10"
  },
  {
    "customer-request-id": 3,
    "logical_clock": 22,
    "interface": "propagate_deposit",
    "comment": "event_recv from branch 2"
  },
  {
    "customer-request-id": 4,
    "logical_clock": 27,
    "interface": "propagate_withdraw",
    "comment": "event_recv from branch 2"
  },
  {
    "customer-request-id": 5,
    "logical_clock": 31,
    "interface": "propagate_deposit",
    "comment": "event_recv from branch 3"
  },
  {
    "customer-request-id": 6,
    "logical_clock": 41,
    "interface": "propagate_withdraw",
    "comment": "event_recv from branch 3"
  },
```

```
{
  "customer-request-id": 7,
  "logical_clock": 46,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 4"
},
{
  "customer-request-id": 8,
  "logical_clock": 56,
  "interface": "propagate_withdraw",
  "comment": "event_recv from branch 4"
},
{
  "customer-request-id": 9,
  "logical_clock": 62,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 5"
},
{
  "customer-request-id": 10,
  "logical_clock": 72,
  "interface": "propagate_withdraw",
  "comment": "event_recv from branch 5"
},
{
  "customer-request-id": 11,
  "logical_clock": 79,
```

```
    "interface": "propagate_deposit",
    "comment": "event_recv from branch 6"
  },
  {
    "customer-request-id": 12,
    "logical_clock": 89,
    "interface": "propagate_withdraw",
    "comment": "event_recv from branch 6"
  },
  {
    "customer-request-id": 13,
    "logical_clock": 97,
    "interface": "propagate_deposit",
    "comment": "event_recv from branch 7"
  },
  {
    "customer-request-id": 14,
    "logical_clock": 107,
    "interface": "propagate_withdraw",
    "comment": "event_recv from branch 7"
  },
  {
    "customer-request-id": 15,
    "logical_clock": 116,
    "interface": "propagate_deposit",
    "comment": "event_recv from branch 8"
  },
```

```
{
  "customer-request-id": 16,
  "logical_clock": 126,
  "interface": "propagate_withdraw",
  "comment": "event_recv from branch 8"
},
{
  "customer-request-id": 17,
  "logical_clock": 136,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 9"
},
{
  "customer-request-id": 18,
  "logical_clock": 146,
  "interface": "propagate_withdraw",
  "comment": "event_recv from branch 9"
},
{
  "customer-request-id": 19,
  "logical_clock": 157,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 10"
},
{
  "customer-request-id": 20,
  "logical_clock": 167,
```

```
    "interface": "propagate_withdraw",
    "comment": "event_recv from branch 10"
  }
 ]
}
```


The combined_events.json file is as follows (only 1 customer-request-id is shown for brevity):


```
[
 {
  "customer-request-id": 1,
  "logical_clock": 1,
  "interface": "deposit",
  "comment": "event_sent from customer 1",
  "id": 1,
  "type": "customer"
 },
 {
  "customer-request-id": 1,
  "logical_clock": 2,
  "interface": "deposit",
  "comment": "event_recv from customer 1",
  "id": 1,
  "type": "branch"
 },
 {
  "customer-request-id": 1,
  "logical_clock": 3,
```

```
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 2",
  "id": 1,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 4,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 3",
  "id": 1,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 5,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 4",
  "id": 1,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 6,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 5",
  "id": 1,
```

```
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 7,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 6",
  "id": 1,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 8,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 7",
  "id": 1,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 9,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 8",
  "id": 1,
  "type": "branch"
},
{
```

```
  "customer-request-id": 1,
  "logical_clock": 10,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 9",
  "id": 1,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 11,
  "interface": "propagate_deposit",
  "comment": "event_sent to branch 10",
  "id": 1,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 4,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 1",
  "id": 2,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 5,
  "interface": "propagate_deposit",
```

```
  "comment": "event_recv from branch 1",
  "id": 3,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 6,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 1",
  "id": 4,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 7,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 1",
  "id": 5,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 8,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 1",
  "id": 6,
  "type": "branch"
```

```
},
{
  "customer-request-id": 1,
  "logical_clock": 9,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 1",
  "id": 7,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 10,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 1",
  "id": 8,
  "type": "branch"
},
{
  "customer-request-id": 1,
  "logical_clock": 11,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 1",
  "id": 9,
  "type": "branch"
},
{
  "customer-request-id": 1,
```

```
  "logical_clock": 12,
  "interface": "propagate_deposit",
  "comment": "event_recv from branch 1",
  "id": 10,
  "type": "branch"
}
```


The implementation results include three parts in 3 different output files:

- All the events with their logical timestamps for each customer in customer_events.json file.
- All the events with their logical timestamps for each branch in branch_events.json file.
- All the events triggered by each customer's deposit/withdrawal requests in combined_events.json.

The justifications for these results are as follows:

- The logical timestamps provide a consistent and ordered view of events in the system, helping to track the execution flow.
- The events are listed in order of their logical timestamps, ensuring the correct order of execution.
- The happens-before relationship is maintained, allowing for accurate coordination and verification of events.

The output format facilitates the analysis of the distributed banking system's behavior and the correct enforcement of Lamport's logical clock algorithm.