IMPROVING PERFORMANCE AND SECURITY IN BYZANTINE FAULT TOLERANT SYSTEMS

CECS 546 – Sarveshwaran Sampathkumar (017387654)

TABLE OF CONTENTS

- Byzantine Fault Tolerance
- Byzantine General Problem
- Byzantine Failure
- Achieving Byzantine Fault Tolerance
- IEEE Reference Paper
- Existing vs Implemented System
- Demo

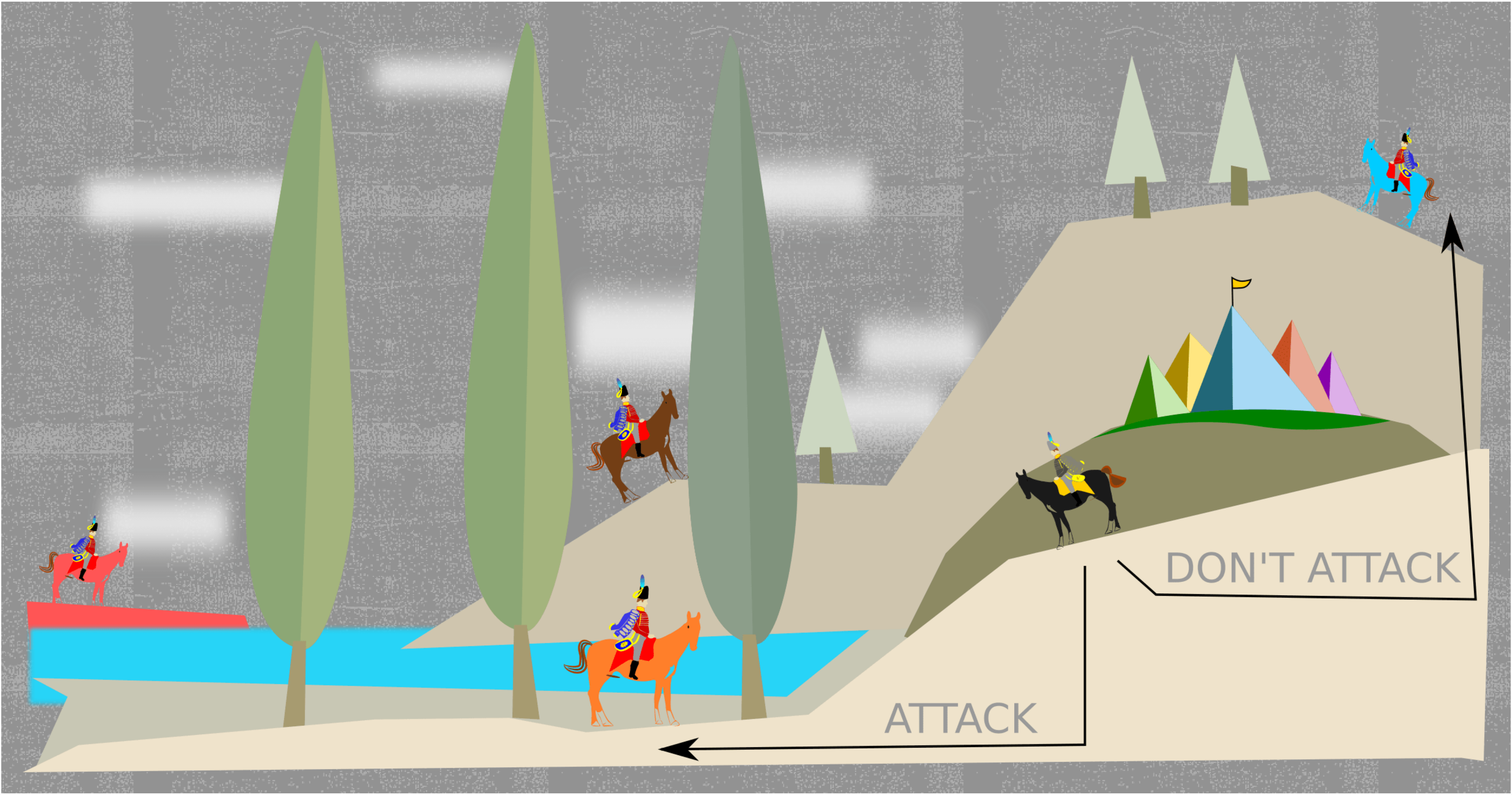
BYZANTINE FAULT TOLERANCE

- In distributed computer systems, Byzantine Fault Tolerance is a characteristic of a system that tolerates the class of failures known as the Byzantine Generals' Problem.

WHAT IS BYZANTINE GENERALS' PROBLEM ?

BYZANTINE GENERALS' PROBLEM

- Several divisions of the Byzantine army are camped outside an enemy city.
- Each division commanded by its own general.
- The generals can communicate with one another only by messenger.
- Must decide upon a common plan of action.


DON'T ATTACK
ATTACK

IDEA !!

- It's impossible to know which generals are traitors trying to prevent the loyal generals from reaching agreement
- The generals must have an algorithm to guarantee that all loyal generals decide upon the same plan of action
- And a small number of traitors cannot cause the loyal generals to adopt a bad plan.

WHAT IS BYZANTINE FAILURE?

- Computers are the generals
- Digital communication system links are the messengers.
- **A Byzantine Fault is a fault that presents different symptoms to different observers**.
- Similarly, **a Byzantine Failure is the loss of a system component due to a Byzantine Fault** in a distributed system that requires consensus.
- Hence, the objective of a **Byzantine Fault Tolerant system is to be able to defend against Byzantine failures**.

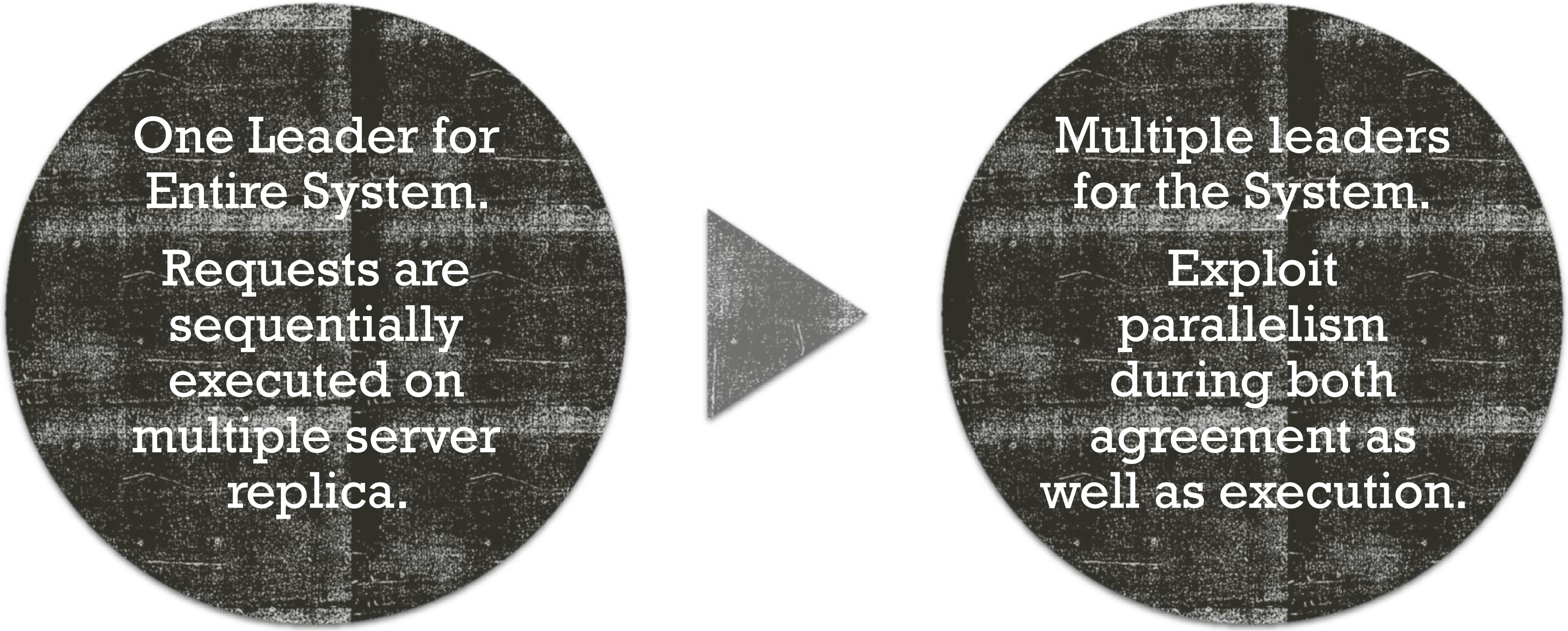
ACHIEVING BYZANTINE FAULT TOLERANCE

- There are 2 prominent solutions that these systems may end up implementing:
- **Unforgeable message signatures**. This may be achieved by using Public-Key Cryptography.
- **Atomic Broadcasts**. If the message system is such that the command is transmitted simultaneously to all participants, then A cannot send a different message to C and B.

REFERENCE IEEE PAPER

- SAREK: Optimistic Parallel Ordering in Byzantine Fault Tolerance
- By Bijun Li, Wenbo Xu, Muhammad Zeeshan Abid, Tobias Distler and Rudiger Kapitza
- At 12th European Dependable Computing Conference - 2016

EXISTING VS IMPLEMENTED SYSTEM



One Leader for Entire System.

Requests are sequentially executed on multiple server replica.

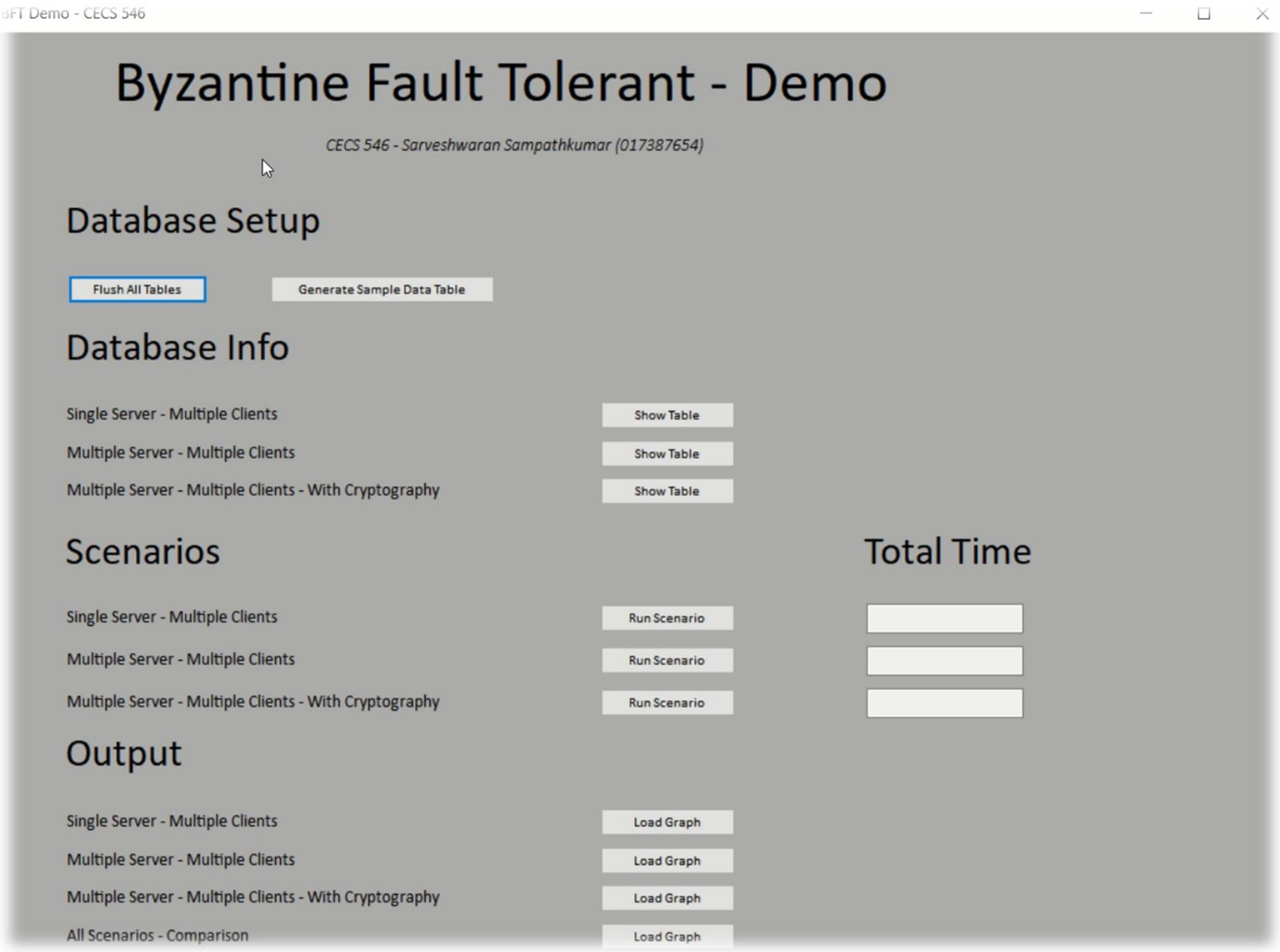
Multiple leaders for the System.

Exploit parallelism during both agreement as well as execution.

IMPLEMENTED SYSTEM

- Aim: Distribute the workload over all the replicas.
- Enables parallelism at the execution stage.
- Existing System – Single leader processes all the requests from client.
- SAREK – Run multiple BFT agreement instances in parallel
- There will be multiple main servers which can act as head.
- When client requests for the information from the server, the serving server, uses a watch dog function whose job is to compare the result of the output from its own server with the other replica servers.
- If the results are the same, then the serving server, responds to the clients request.
- If the results are different, the server which returned the correct output will be made the head and the that head server would serve the client request.
- Encryption and Decryption are handled by RSA Algorithm.

DEMO



BFT Demo - CECS 546

Byzantine Fault Tolerant - Demo

CECS 546 - Sarveshwaran Sampathkumar (017387654)

Database Setup

Database Info

Single Server - Multiple Clients

Multiple Server - Multiple Clients

Multiple Server - Multiple Clients - With Cryptography

Scenarios

Total Time

Single Server - Multiple Clients

Multiple Server - Multiple Clients

Multiple Server - Multiple Clients - With Cryptography

Output

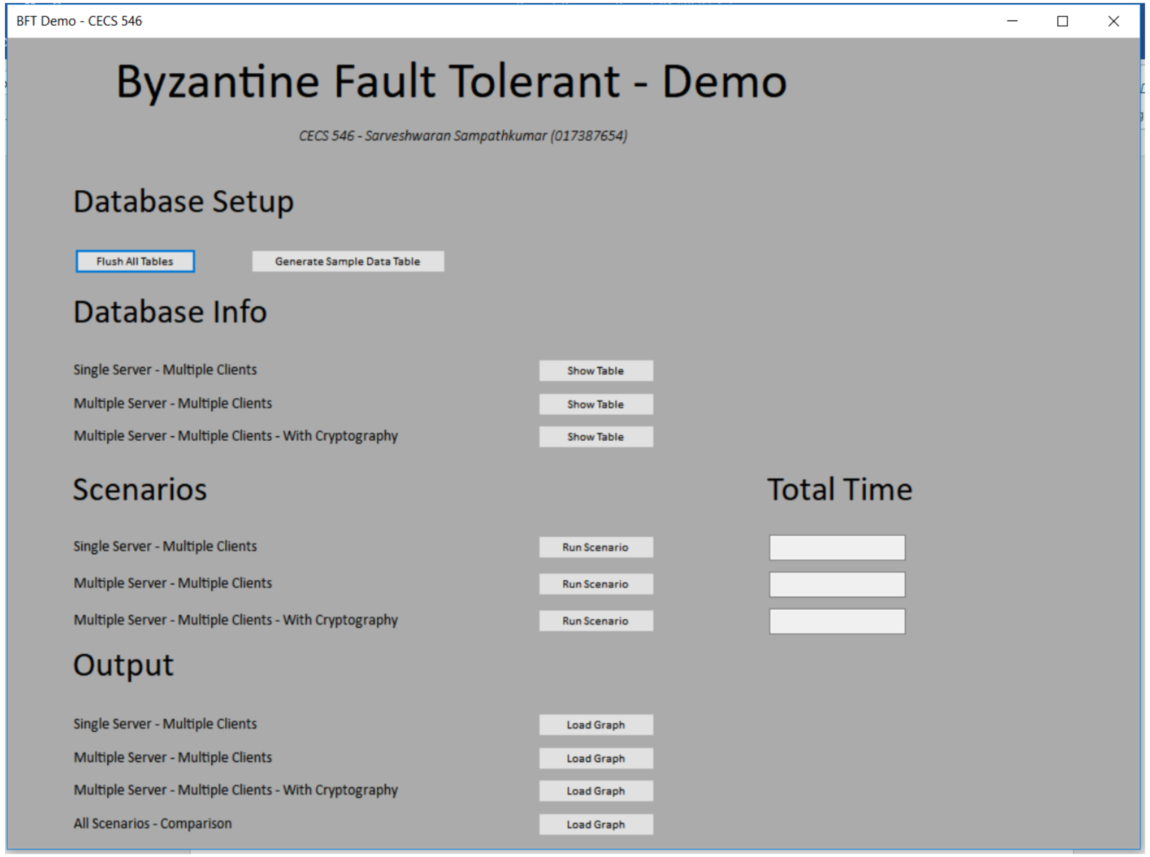
Single Server - Multiple Clients

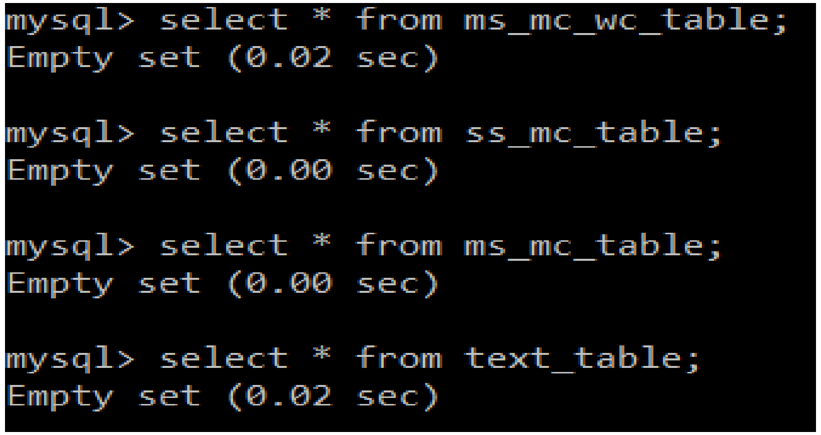
Multiple Server - Multiple Clients

Multiple Server - Multiple Clients - With Cryptography

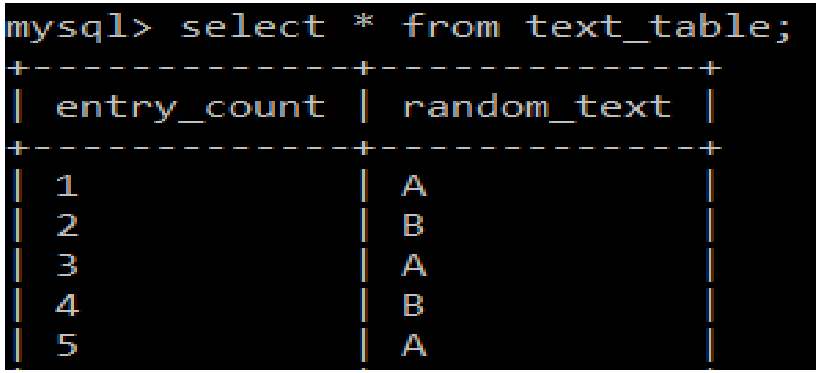
All Scenarios - Comparison

OUTPUT


BFT Demo - CECS 546
Byzantine Fault Tolerant - Demo
CECS 546 - Sarveshwaran Sampathkumar (017387654)
Database Setup
Flush All Tables
Generate Sample Data Table
Database Info
Single Server - Multiple Clients
Show Table
Multiple Server - Multiple Clients
Show Table
Multiple Server - Multiple Clients - With Cryptography
Show Table
Scenarios
Total Time
Single Server - Multiple Clients
Run Scenario
Multiple Server - Multiple Clients
Run Scenario
Multiple Server - Multiple Clients - With Cryptography
Run Scenario
Output
Single Server - Multiple Clients
Load Graph
Multiple Server - Multiple Clients
Load Graph
Multiple Server - Multiple Clients - With Cryptography
Load Graph
All Scenarios - Comparison
Load Graph

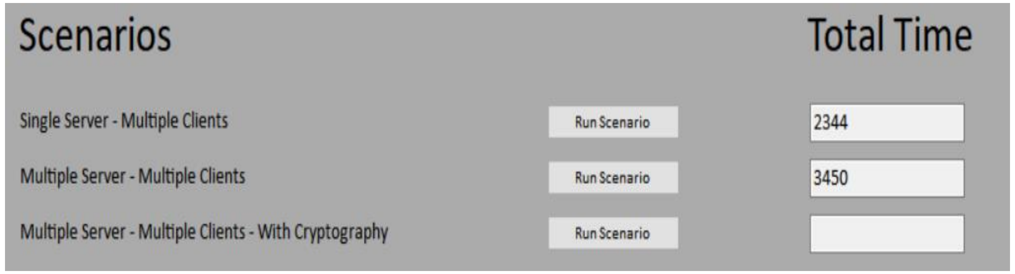


```
mysql> select * from ms_mc_wc_table;
Empty set (0.02 sec)

mysql> select * from ss_mc_table;
Empty set (0.00 sec)

mysql> select * from ms_mc_table;
Empty set (0.00 sec)

mysql> select * from text_table;
Empty set (0.02 sec)
```



```
mysql> select * from text_table;
+-------------+-------------+
| entry_count | random_text |
+-------------+-------------+
| 1           | A           |
| 2           | B           |
| 3           | A           |
| 4           | B           |
| 5           | A           |
```

OUTPUT

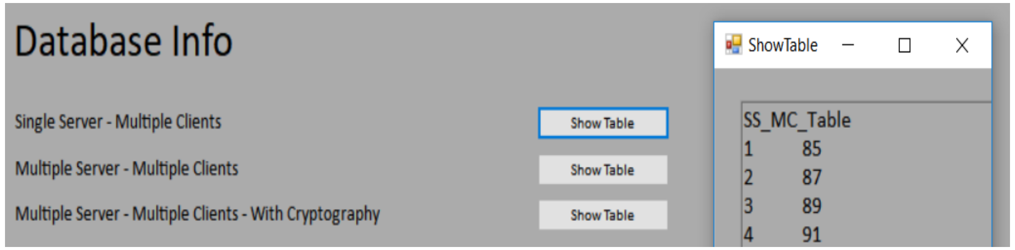


Scenarios | Total Time

Single Server - Multiple Clients | Run Scenario | 2344

Multiple Server - Multiple Clients | Run Scenario | 3450

Multiple Server - Multiple Clients - With Cryptography | Run Scenario |



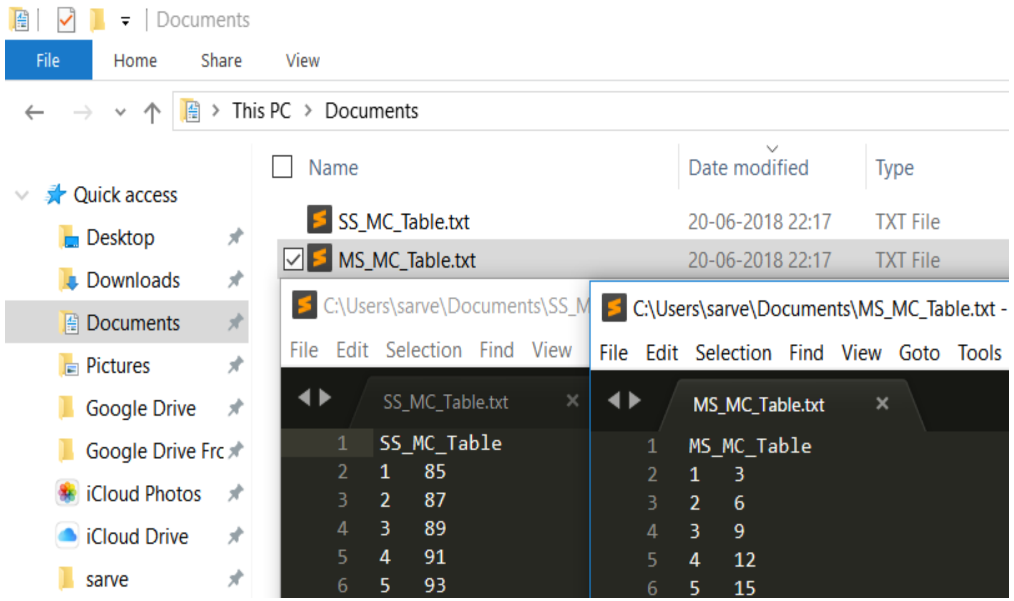
Database Info

Single Server - Multiple Clients | Show Table

Multiple Server - Multiple Clients | Show Table

Multiple Server - Multiple Clients - With Cryptography | Show Table

```
SS_MC_Table
1       85
2       87
3       89
4       91
```



Name	Date modified	Type
SS_MC_Table.txt	20-06-2018 22:17	TXT File
☑ MS_MC_Table.txt	20-06-2018 22:17	TXT File

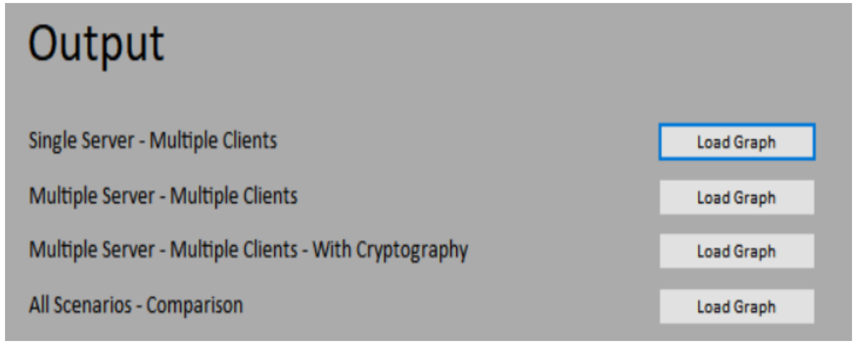
C:\Users\sarve\Documents\SS_M

```
SS_MC_Table
1   85
2   87
3   89
4   91
5   93
```

C:\Users\sarve\Documents\MS_MC_Table.txt -

```
MS_MC_Table
1   3
2   6
3   9
4   12
5   15
```

OUTPUT



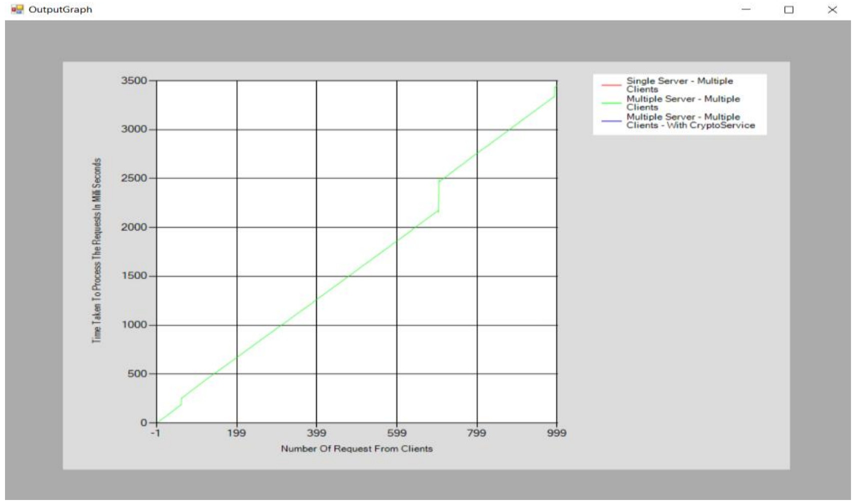
Output

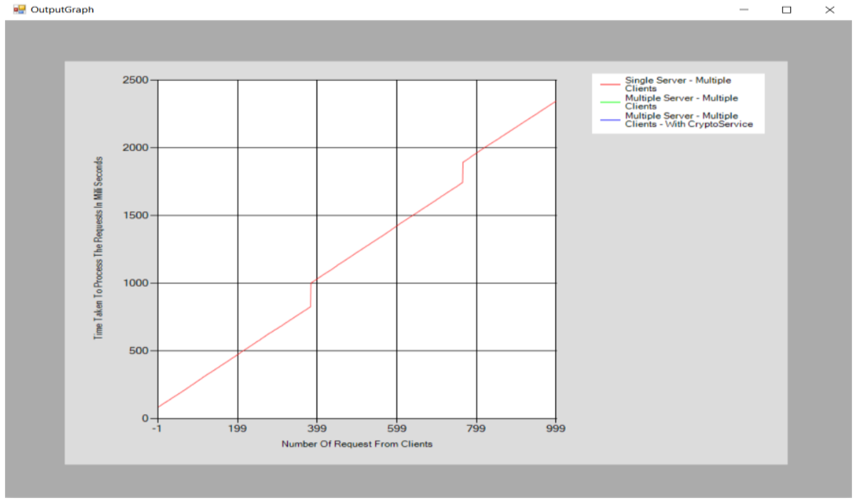
Single Server - Multiple Clients — Load Graph

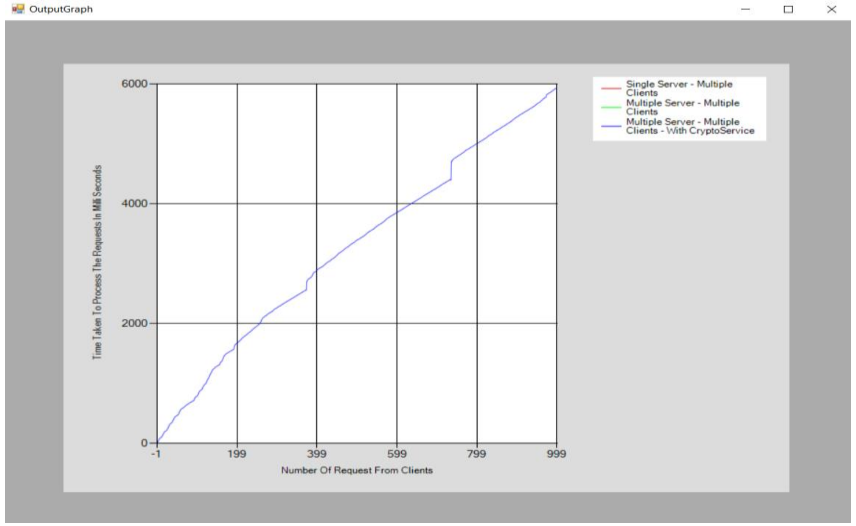
Multiple Server - Multiple Clients — Load Graph

Multiple Server - Multiple Clients - With Cryptography — Load Graph

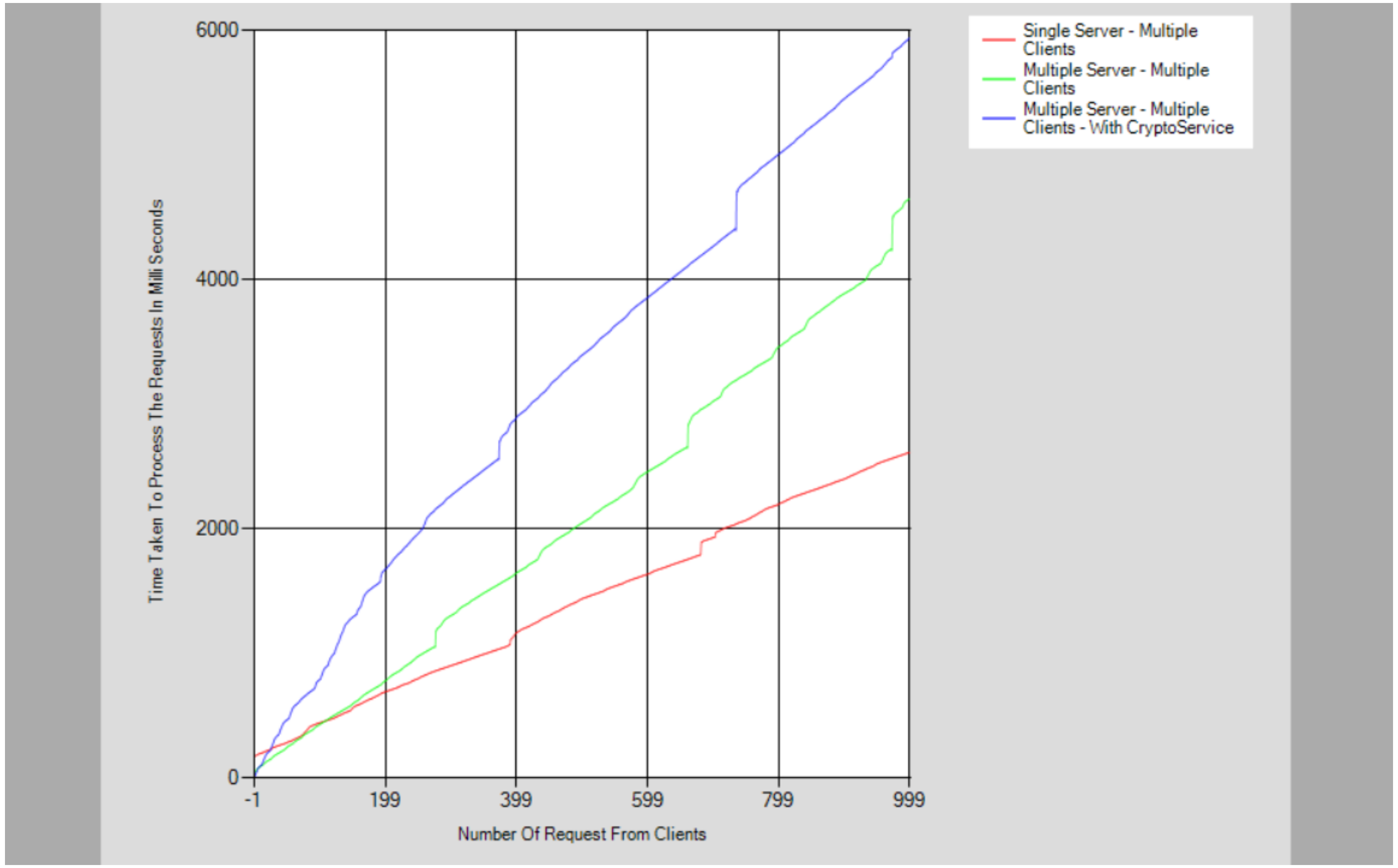
All Scenarios - Comparison — Load Graph







COMPARISON



Single Server - Multiple Clients
Multiple Server - Multiple Clients
Multiple Server - Multiple Clients - With CryptoService
6000
4000
2000
0
-1
199
399
599
799
999
Time Taken To Process The Requests In Milli Seconds
Number Of Request From Clients

Q & A SESSION

REFERENCES:

- Presentation: https://blog.cdemi.io/byzantine-fault-tolerance/
- Further Reading on Algorithms and Proofs: https://cs.uwaterloo.ca/~tozsu/courses/CS755/F13/Presentations/jose.pdf
- IEEE Reference Paper: SAREK: Optimistic Parallel Ordering in Byzantine Fault Tolerance – 2016 – By Bijun Li, Wenbo Xu, Muhammad Zeeshan Abid, Tobias Distler and Rudiger Kapitza at 12th European Dependable Computing Conference. https://ieeexplore-ieee-org.csulb.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=7780347