# Centralized Computing System (CCS) Documentation

## Student Number: s31722

### 1. Implemented Features

#### 1.1 Service Discovery

- Functionality:

- Listens for UDP messages on the specified port.

- Responds to discovery messages (`CCS DISCOVER`) with `CCS FOUND`.

- Resumes listening after sending the response.

- Implementation:

- The method `handleDiscovery` manages UDP communication and ensures proper responses to discovery messages.

#### 1.2 Communication with Clients

- Functionality:

- Opens a TCP socket for client connections on the specified port.

- Handles multiple concurrent clients using a thread pool.

- Accepts requests in the format `<OPER> <ARG1> <ARG2>`.

- Performs operations (ADD, SUB, MUL, DIV) and returns the result.

- Responds with `ERROR` for invalid requests (e.g., division by zero, malformed input).

- Implementation:

- The methods `handleClients` and `handleClient` implement TCP communication.

- A thread pool (`ExecutorService`) ensures scalability for multiple clients.

- Comprehensive error handling is included for invalid operations and arguments.

#### 1.3 Statistics Reporting

- Functionality:

- Tracks the following statistics:

- Total number of connected clients.

- Number of processed requests.

- Count of specific operations (ADD, SUB, MUL, DIV).

- Number of invalid requests.

- Sum of all results.

- Prints global and interval statistics every 10 seconds.

- Implementation:

- The method `printStatistics` manages statistics collection and periodic reporting.

- Atomic variables ensure thread-safe updates to counters and sums.

### 2. Unimplemented Features

- Retry Mechanism for Lost UDP Packets:

- The current implementation does not include a mechanism to detect or resend lost UDP packets.

- Advanced Error Recovery:

- If a client sends a corrupted request, the program handles it gracefully but does not attempt advanced recovery.

### 3. Difficulties Encountered

- Concurrency:

- Managing synchronization between threads for shared statistics was challenging but resolved using atomic variables.

- UDP Communication:

- Ensuring consistent response handling and debugging malformed discovery requests required additional testing.

- Testing Multiple Clients:

- Simulating multiple concurrent clients required careful orchestration and stress testing to validate server performance.

### 4. Remaining Errors and Limitations

- UDP Packet Loss:

- No guarantee of packet delivery due to the nature of UDP.

- Statistics Delay:

- The 10-second interval may not capture rapid spikes in activity in real time.

- Client Disconnection:

- If a client disconnects unexpectedly, the server gracefully handles it, but the disconnection is only logged.

### 5. Designed and Implemented Protocol

#### 5.1 Discovery Protocol

- Message Format:

- Request: `CCS DISCOVER`

- Response: `CCS FOUND`

- Communication:

- Client sends a UDP broadcast; server responds with a unicast message.

#### 5.2 Client Communication Protocol

- Request Format: `<OPER> <ARG1> <ARG2>`

- `OPER`: One of ADD, SUB, MUL, DIV.

- `ARG1`, `ARG2`: Integer arguments.

- Response Format:

- Success: Single integer (result).

- Error: `ERROR` string for invalid requests.

### 6. Example Execution

#### Service Discovery

- Command: `java CCS <port>`

- Client Sends: `CCS DISCOVER` (UDP)

- Server Responds: `CCS FOUND` (UDP)

#### Client Communication

- Command: `java CCS <port>`

- Client Sends: `ADD 5 10` (TCP)

- Server Responds: `15`

### 7. Conclusion

The CCS application successfully implements the core functionalities required by the task. It supports service discovery, client communication, and statistics reporting while handling multiple clients concurrently. Despite limitations inherent in the UDP protocol and basic error handling, the system is robust and meets the majority of the requirements effectively.