Remote Execution Framework
OMNeT++ Simulation Project

B22CS047 & B22CS035

April 1, 2025

1 Project Overview

This project implements a distributed remote execution framework using OMNeT++. The simulation models a network of client and server nodes, where clients can send computation tasks to servers for processing. The framework handles task distribution, result collection, and supports various task types.

2 File System Structure

The project consists of the following files:

```
RemoteExecution/
|-- topo.txt # Network topology configuration
|-- RemoteMessages.msg # Message definitions
|-- RemoteExecution.ned # Network description
|-- ServerNode.cc # Server implementation
|-- ClientNode.cc # Client implementation
|-- NetworkBuilder.cc # Dynamic network builder
|-- omnetpp.ini # Simulation configuration
```

3 File Descriptions

3.1 topo.txt

This file defines the network topology, specifying the number of clients and servers and their connections.

```
Format: <numClients> <numServers>
3 5

Client connections (optional)
0 1,2
1 0,2
2 0,1
```

3.2 RemoteMessages.msg

Defines the message types used for communication between clients and servers.

```
// Message definitions for remote execution
packet TaskMessage {
    int taskId;
    string taskType;
    int clientId;
    int numElements;
    int elements[];
}

packet ResultMessage {
    int taskId;
    int clientId;
    int result;
    simtime_t processingTime;
}
```

3.3 RemoteExecution.ned

Network description file that defines the structure of the simulation, including module types, parameters, gates, and connections.

```
package RemoteExecution;

simple ServerNode {
    parameters:
        int nodeId;
        double maliciousProbability = default(0.2);
    gates:
        input in[];
        output out[];
}

simple ClientNode {
    parameters:
        int nodeId;
        string taskType = default("findMax");
        int numElements = default(100);
        int elementMin = default(0);
        int elementMax = default(1000);
    gates:
        input in[];
        output out[];
}

network RemoteExecutionNetwork {
    parameters:
        int numClients @prompt("Number of clients");
        int numServers @prompt("Number of servers");
    submodules:
        client[numClients]: ClientNode;
        server[numServers]: ServerNode;
    connections allowunconnected;
}
```

3.4 ServerNode.cc

Implements the server module behavior, including task processing, result generation, and potential malicious behavior.

3.5 ClientNode.cc

Implements the client module behavior, including task generation, server selection, and result verification.

3.6 NetworkBuilder.cc

Dynamically builds the network topology based on the configuration in topo.txt, setting up connections between clients and servers.

3.7 omnetpp.ini

Configuration file for the simulation, specifying parameters and simulation settings.

```
[General]
network = RemoteExecutionNetwork
sim-time-limit = 100s

# Client parameters
**.client[*].numElements = 100
**.client[*].elementMin = 0
**.client[*].elementMax = 1000
**.client[*].taskType = "findMax"

# Server parameters
**.server[*].maliciousProbability = 0.2

# Visualization settings
**.vector-recording = true
**.scalar-recording = true
```

4 Building and Running the Simulation

4.1 Prerequisites

- OMNeT++ 6.0 or later
- C++ compiler (GCC 7.0+ or Clang)

4.2 Build Instructions

1. Open the project in OMNeT++ IDE
2. Build the project (Project → Build Project)
3. Ensure topo.txt is in the project root directory

4.3 Running the Simulation

1. Right-click on the project → Run As → OMNeT++ Simulation
2. Enter the number of clients and servers when prompted
3. Use the simulation controls to run the simulation

5 Simulation Features

- Dynamic network topology creation
- Multiple task types (findMax, findMin, sum, average)
- Malicious server detection and handling
- Performance metrics collection
- Task distribution and load balancing

6 Troubleshooting

- If you encounter "Gate size is 0" errors, ensure NetworkBuilder.cc properly sets gate sizes before establishing connections
- For package declaration errors, verify that the package name in RemoteExecution.ned matches the directory structure
- If topo.txt is not found, check that it's in the correct location and properly formatted