Documentation for Distributed Averaging System (DAS) Project

Introduction

The **Distributed Averaging System (DAS)** project implements a UDP-based communication system that can dynamically operate in either **Master Mode** or **Slave Mode**. The mode is determined automatically based on the system's current state. The application demonstrates the ability to collect and process distributed numerical data, compute averages, and broadcast messages across a local network.

This project adheres to the specification provided, using Java 8 (JDK 1.8), and has been designed and implemented with basic UDP socket classes.

Overview of Application Behavior

Application Modes

The application can operate in the following two modes:

1. **Master Mode**:
    - Activates if the application successfully opens the requested UDP port.
    - Collects numerical data from Slaves.
    - Computes and broadcasts the average of received non-zero numbers.
    - Terminates upon receiving a -1 message.
2. **Slave Mode**:
    - Activates if the requested UDP port is already in use.
    - Sends a single number (specified by the user) to the Master operating on the specified port.
    - Terminates after sending the message.

Command Syntax

The application is executed with the following command:

java DAS <port> <number>

- <port>: UDP port number to be used by the Master or contacted by the Slave.
- <number>: An integer value to be stored or sent, depending on the mode.

Design and Communication Protocol

Communication Protocol

1. **Messages Sent by Slave**:
    - A single UDP message containing the <number> parameter.
    - Destination: localhost at the specified <port>.
2. **Messages Handled by Master**:
    - **Non-Zero Positive Numbers**: Stored and printed to the console.

- **Zero**:
    - Computes the average of all stored non-zero numbers.
    - Broadcasts the computed average to the local network on the same <port>.
- **Negative One (-1)**:
    - Signals termination.
    - Broadcasts -1 to the network.
    - Terminates the Master process.

3. **Broadcasting by Master**:
    - Sends UDP packets to all reachable machines in the local network (via broadcast addresses).

Design Considerations

- **Automatic Mode Selection**: The application uses the ability to open the requested port to decide whether to operate as Master or Slave.
- **Concurrent Processing**: The Master handles incoming messages in a loop while maintaining stored data.

Class Descriptions

DAS

Purpose: The entry point of the application.

- Parses and validates command-line arguments.
- Initializes DASPresenter for application logic.
- Decides operational mode (Master/Slave).

DASPresenter

Purpose: Coordinates interaction between the user interface and the core networking logic.

- Delegates mode-specific operations to Master or Slave.
- Handles callback messages for UI updates.

Client

Purpose: A base class for networking entities (Master and Slave).

- Stores the port and number attributes.
- Provides basic access methods.

Master

Purpose: Implements the logic for the Master mode.

- Opens the specified port for UDP communication.
- Handles incoming messages based on their content (0, -1, or other numbers).
- Computes averages and broadcasts data to the local network.
- Maintains a synchronized list of all received numbers.

Slave

Purpose: Implements the logic for the Slave mode.

- Sends the specified <number> to the Master at the given port.
- Uses a randomly assigned UDP port for communication.
- Terminates immediately after sending the message.

ConsoleView

Purpose: Provides methods to display messages and errors to the console.

- Used by DASPresenter to output application feedback.

How to Run the Application

Challenges Faced

1. **Broadcast Address Discovery**:
    - Identifying broadcast addresses for local networks required careful usage of Java's NetworkInterface API.
2. **Concurrency**:
    - Ensuring the Master could handle incoming messages while maintaining data consistency required synchronized data structures.