**Implementation of Distributed Averaging System (DAS)**

**Submitted by student: s30175**

**1. Overview**

The Distributed Averaging System (DAS) is a Java-based application designed to operate in two modes: Master and Slave, determined dynamically at runtime. This document details the features implemented, challenges faced, protocol design, and potential improvements.

**Usage**

*java DAS <port> <number>*

* <port>: Specifies the UDP port number for communication.
* <number>: An integer or floating-point value to be processed. Non-integer inputs are rounded down using Math.floor().

**Modes of Operation**

* Master Mode: Initiated if the application successfully binds to the specified <port>. The process listens for incoming messages, processes received numbers, and broadcasts results.
* Slave Mode: Activated if the specified <port> is already in use. The process sends the specified <number> to the Master and terminates.

**2. Implemented Features**

* 1. Master Mode
* Initialization: binds to the specified <port> and stores the <number> parameter.
* Message handling:
  + 0: Computes and broadcasts the average of all non-zero numbers received (including <number>).
  + -1: Broadcasts a termination signal and shuts down.
  + Any other integer: Stores the value for later average computation.
* Broadcasting: Sends computed averages or termination signals to all devices on the local subnet using a calculated broadcast address.
* Input validation: Ensures only valid numbers are processed.
* Self-message filtering: Ignores packets originating from the Master itself.

2.2 Slave Mode

* Initialization: Opens a UDP socket on a random port assigned by the OS.
* Message transmission: Sends the specified <number> to the Master at <port>.
* Termination: Exits immediately after successfully sending the message.

**3. Unimplemented Features**

* Support for IPv6 networks.
* Comprehensive logging for network errors, packet losses, or invalid message formats.

**4. Protocol Description**

4.1 Communication Protocol

* Transport Layer: UDP ensures fast communication but lacks guaranteed delivery.
* Message Format: Plain text integers.

4.2 Protocol Actions

* Master Mode listens for incoming UDP messages and processes integers as follows:
  + 0: Computes the average, prints to the console, and broadcasts it.
  + -1: Prints termination message, broadcasts, and exits.
  + Other Integers: Adds to the stored list and prints to the console.
  + Broadcasts messages using the local network's calculated broadcast address.
* Slave Mode sends <number> to the Master process and terminates.

4.3 Edge Cases

* Broadcast address calculation accounts for NAT and subnet mask variations.
* The program filters malformed messages and logs warnings.
* Slave mode terminates gracefully if Master is unreachable.

**5. Difficulties Encountered**

* Handling dynamic network configuration, subnet masks, and calculating broadcast addresses across diverse environments.
* Differentiating between runtime network issues and logical errors in message processing.
* Ensuring thread-safe broadcasting during average computations.

**6. Existing Errors and Limitations**

* IPv6 is currently unsupported.
* Network issues (e.g., packet loss) are not logged explicitly.
* Extremely large inputs may cause parsing errors.

**7. Conclusions**

The implementation meets the specified requirements, providing a functional Master-Slave communication system using UDP. The project demonstrates key concepts of distributed systems, including message broadcasting, input validation, and error handling. Potential improvements include IPv6 compatibility, robust error logging, and enhanced protocol scalability.