CP372 Assignment 1
Request for Comments:
Group Number: 13

Nicolas Ross
Lucas Krete

Post-it Note Protocol

TABLE OF CONTENTS

1 - Introduction

1.1 Purpose

The Post-It Note Protocol (P.I.N Protocol) will allow multiple users to connect to a server to post, view, delete, and pin/unpin notes of their choosing. The client will be able to establish a connection to the server via a Graphical User Interface (G.U.I). The purpose of the P.I.N Protocol is to gain an understanding of servers and clients and their interactions using a protocol.

1.2 Requirements

The client must have an understandable G.U.I that allows the user to easily send requests to the server. It must also be able to send five types of requests to the server:

"POST <note>" - server must store a note specified under <note>
"GET <request>" - server sends the client all notes that satisfy properties in <request>
"PIN/UNPIN <data>" - server must place, or delete pins corresponding to <data>
"CLEAR" - Server must forget all unpinned notes
"DISCONNECT" - Client will disconnect from the server

The server must handle multiple requests for connection from different clients and allow them all to send requests concurrently. The requests received from the client must be processed and appropriately handled. The server must hold all the data that has been received from the client for the duration of the servers "up-time" and then erase the data when the server is shutdown.

1.3 Terminology

<data>
A pair of integers, describing x and y coordinates of a point on the note board that is within its bounds.

<note>
Similar to a physical post-it note and is defined by the x and y coordinates of the lower-left corner, width, height, color, and message. If a client would like to POST a note, it must be completely defined.

When notes are posted they are not pinned, the client must use the PIN or UNPIN command. Any pin that is placed inside of a note's perimeter will be pinned on placement, or unpinned on removal given the note was not already pinned/unpinned by a preexisting pin.

<request>
Field in the GET command that has three forms; GET PINS (server supplies the client with the coordinates of all the pins), GET ALL (signified by leaving all fields blank and will return the message

contained in all notes), and GET colour= <colour> contains= <data> refersTo= <string> (server supplies the client with all pins of specified fields.)

1.4 Overall Operation

The P.I.N Protocol is a request/response protocol where the client sends a request to the server which will make certain changes to the board contained within the server. The user will connect to the server using the client where they will use a G.U.I to create a request in the fields provided. This request will allow the user to post, get, pin/unpin a note, clear all pinned notes from the server and finally, disconnect from the server. The client will error check the request to ensure the requirements are fulfilled, then sends the request to the server. The server then reads the request to determine which action is to be executed. There is then another check on the server side to determine if the request passes all the requirements and responds accordingly.

2 - Responsibilities

2.1 Server Responsibilities

The server is responsible for accepting a successful connection to the client, as well as be able to maintain/accept connections from different clients simultaneously. The server must be able to take input from the different clients, process these requests, and use the protocol to update our Post it note board (LinkedList), as well as our pin list (ArrayList). The structures needs to persist as long as the server is live. Furthermore, the server must check the data for errors and efficiently communicate with the client when an error is raised. Lastly, the server is responsible for notifying the user when their request is either successful or unsuccessful..

2.2 Client Responsibilities

The client is responsible for requesting the connection to the server, handling all post it note operations, and producing a usable GUI. The client will read user input via text fields and buttons in the GUI, and forward this data to the server for proper storage.
The GUI will contain all required fields for text, as well as buttons for posting, pinning/unpinning, connecting/disconnecting, get and lastly an area to display notes.

3 - Format of Data

3.1 Format of Note Object

The Note object will be in the following format: Object = (Int X-Coordinate, Int Y-Coordinate, Int Width, Int Height, String Colour, String Message, Int Status, ArrayList<Pin> pins), and will be stored in a linked list. The use of the arrayList of pins is to easily access which pins are currently pinning any given note.

3.2 Format of Pin Object

The Pin object will be in the following format: Object = (Int X-Coordinate, Int Y-Coordinate) and will be stored in an ArrayList for easy accessibility.

3.3 Client Requests

Each request will be sent in a single line format based on the instruction given as the first string of the input. The user will choose the instruction then input the necessary data for the request:

POST - <Int x-coordinate> < Int y-coordinate> <Int width> <Int height> <String colour> <String message>

GET - <String colour> <Int x-coordinate> <Int y-coordinate> <String message>
GET - “ALL”
GET - “PINS”

PIN/UNPIN - <Int x-coordinate> <Int y-coordinate>

CLEAR - “CLEAR”

DISCONNECT - “DISCONNECT”

3.4 Server Responses

After a request from the client, the server will respond with a notification letting the user know the request was completed. For POST, PIN, UNPIN, and CLEAR requests, the server will display a message notifying the user if the note was successfully posted, pinned, unpinned, or all of the unpinned notes were cleared respectively. When the user requests the GET instruction the server responds with all the notes that match the users specific request. For example, if the user only inputs the colour field as “Blue” then it will return all notes with the specified colour.

4 - Synchronization Policies

While the server will be multi-threaded by nature, requests to the server from it’s corresponding clients will be processed on a first-come-first-serve basis. This is to avoid any conflict in processing two requests simultaneously, as well to improve turnaround times. For example, if two requests are sent to the server, one to post, and one to pin, whichever is received first is the command that will be executed first. Therefore, any request that is received by any amount of time before another, will be executed first.

5 - Protocol Parameters

The Post-it Note Protocol will take the information provided by the client; Pin/Unpin, Post, Get and will process the data that is received through the GUI. It will then determine which operation to perform, and interact with the server accordingly.

6 - Data Structure

The data structure used to store the notes on the server will be a linked list of note-nodes. This is used to allow for dynamic memory allocation to help manage memory, as well as simplicity. This will allow for easy linear searching while also making the structure robust and reliable. The structure will be filled with nodes (notes objects) but will be erased once the server is shut down. The note object will hold: X-Coordinate, Y-Coordinate, Width, Height, Colour, Message, Status, and pins.

7 - Errors

7.1 Client Errors

Incomplete Fields
- Displays error to user, allows them to fill in required fields

Connection Failed
- Displays error to user, reroutes back to IP/port page

Incorrect IP/Port
- Displays error to user, allows the user to fix their error

7.2 Server Errors

Not using client
- Disconnects from client

Connection Failed

7.3 Border Cases

1. Incorrect coordinates:
    a. If user inputs negative value or a value out of bounds, sends message to user notifying of a possible incorrect coordinate
2. First client requests GET without posting anything:
    a. Displays message notifying user that there are no posts
3. Server shuts down unexpectedly:
    a. Display message that the server is unresponsive
4. Unpin request for non-existent pin:
    a. Display message that the pin does not exist
5. Clear notes when there are no unpinned notes
    a. Display message to user saying there are no unpinned notes to clear

8 - Security

To prevent unauthorized access of the server, implemented, is a client-server password. Both the server and client hold the same password and the server can only be accessed once that password is verified on both ends. This ensures that users aren't able to remotely access our server. Upon startup of the client, it will request access to the server by validating the password that both server and client possess. If the passwords match (user is on the client) access will be granted, otherwise access will be denied and the server will decline access. This is to ensure that full control of what is being sent to the server is being achieved.