SwapBytes Assignment Report

Course: COSC 473 – Decentralised Applications on the Web
Assignment: Assignment 2 - SwapBytes
Author/Student: Haig Bishop
Repository: https://github.com/HaigBishop/swapbytes

1. Introduction

This report describes SwapBytes I developed for Assignment 2 of COSC 473. My goal was to create a visually pleasing and easy-to-use CLI/TUI for P2P file transfer and chats.

The resulting application utilised `ratatui` for the TUI, which displays a "Users" pane listing all active users, a "Chat" pane which enables both global and private chats, and a "Console" pane which allows to user to run commands. P2P features such as connectivity, messaging and file transfer are implemented using `libp2p`.

2. How SwapBytes Works

SwapBytes allows users to connect, chat and transfer files in a decentralised manner. It leverages multiple tools and techniques:

- TUI - All user interaction goes through the terminal user interface powered by `ratatui`.
  - The `App` struct in `tui.rs` implements `ratatui::widgets::Widget` and holds the entire TUI state.
  - The terminal area is divided into three main panes: Chat, Console, and Users. Each rendered seperately according to the state of `App`.
  - Keyboard events are captured by a seperate task in `main.rs` and sent to `handle_key_event` in `input_handler.rs` ultimately modifiying the state of `App`.
- mDNS - Peers on the same device and/or network can connect automatically via Multicast DNS.
  - In `behavior.rs`, the `SwapBytesBehaviour` struct includes an `mdns: mdns::tokio::Behaviour`.
  - This behaviour automatically discovers other peers on the local network also using mDNS.
  - It adds discovered peers as explicit peers to the Gossipsub. When peers expire or are no longer discovered via mDNS, they are removed from Gossipsub.
  - All other users are displayed on the Users list along with whether or not they are online.
- Ping - Users can manually ping other users to initiate stable connections.
  - The `SwapBytesBehaviour` struct also includes `ping: ping::Behaviour`.
  - The pinging is configured to have a very long interval (`Duration::from_secs(3 * 60 * 60)`) effectively disabling automatic pings.
  - Instead, the purpose of pings in SwapBytes is the `/ping <multiaddr>` command. This gives users the ability to initiate stable connections with particular peers.
- Rendezvous - A rendezvous server allows peers on different networks to connect.
  - The `SwapBytesBehaviour` struct also includes `rendezvous: rendezvous::client::Behaviour`.
  - Constants `RENDEZVOUS_PEER_ID`, `RENDEZVOUS_ADDR`, and `RENDEZVOUS_NS` (namespace) are defined in `constants.rs`.
  - At start-up peers automatically dial the `RENDEZVOUS_ADDR` and if they connect, they register with the server in the `RENDEZVOUS_NS` namespace.
  - Periodically, peers request a list of other peers registered in the same namespace from the rendezvous.
  - The document `how-to-rendezvous.md` on the GitHub repo contains clear instructions on how to set-up a server.
- Gossipsub - A gossipsub topic enables all general communication such as heartbeats, global chats, and nickname updates.
  - The `SwapBytesBehaviour` struct also includes `gossipsub: gossipsub::Behaviour`.
  - A single topic, `constants::SWAPBYTES_TOPIC`, is used for heartbeats and global chat messages.

    - Upon recieving messages the message data is deserialized from JSON into `protocol::Message` and handled accordingly. e.g. if it is a heartbeat, the sender's nickname and `last_seen` are updated. Or if it is a global chat, the actual text contents are stored in `app.global_chat_history` and displayed on the TUI.
- Request Response - Direct P2P communications (private messages, file offers and file transfers) are handled by a custom request-response protocol.
    - The protocol defined in `protocol.rs` including `PrivateRequest` and `PrivateResponse` enums containing message types (e.g. `ChatMessage`, `Ack`, `Offer`, `AcceptOffer`, `RequestChunk`, and `FileChunk`,)
    - The `SwapBytesBehaviour` struct includes a `request_response::Behaviour<PrivateCodec>` with a 2 MiB `LengthDelimitedCodec` for request response in SwapBytes.
    - Upon recieving a `PrivateRequest` (e.g. `ChatMessage`) typically the UI is updated and a response is sent to the requesting peer with a `Ack` (acknowledge).
    - The request-response structure for file transfer starts with a `PrivateRequest::Offer`. If the recipient accepts (via the `/accept` command) they send a `PrivateRequest::AcceptOffer` and immediately begins the download by sending the first `PrivateRequest::RequestChunk`. When the peer who sent the offer recieves the `AcceptOffer` all following `RequestChunk` requests are responded to with a `PrivateResponse::FileChunk` until all chunks are sent or termination by `PrivateResponse::TransferError`.

For more practical information on how SwapBytes works, see `README.md` and `demo.md` on the GitHub repo.

3. Challenges and Solutions

Challenge 1: Who's Who?

- Problem: After implementing nickname propogation by adding a `nickname` field to heartbeat gossipsub messages, I noticed that if there were > 2 users online at once user's nicknames frequently flip-flopped (e.g. 'Bob' -> 'Alice' -> 'Bob', etc.)
- Cause: In short, using `propagation_source` to update nicknames led to the nickname of the original sender being temporarily attributed to the *forwarding peer* and not the peer who initially created and signed it. This would have caused messages to also be miss-attributed, but at this stage I was only using Gossipsub for nicknames and online status.
- Solution: I discovered that the `gossipsub::Message` struct contains a `source: Option<PeerId>` field, which holds the PeerId of the *original* sender. This `source` (and not the `propagation_source`) was used to correctly associate the nickname (and other message content) with the PeerId of the actual sender.

Challenge 2: They see me scrolling.

- Problem: By default the scrollable panes from the `ratatui` TUI scrolled really far down, so far down that the final line was at the top and there was a lot of empty space below it.
- Cause: I was setting the scroll position to the number of lines in the content. But since `ratatui` interperets this as the number of lines to hide from the top, this positioned the final line of the content at the top of the pane.
- Solution: Along with other custom scrolling logic, I subtracted the height of the viewport from updated scroll positions to get the behavior how I wanted it.

4. Requirements & Bonus Features

- Basic Requirements: All basic requirements have been met:
    - CLI/TUI interface (implemented with `ratatui` TUI).
    - Users get auto-assigned usernames, then can change them with `/setname`.
    - There is a global chat where all users can see all messages.
    - Direct message chats can be opened to send/receive private messages.
    - File sending via request/response (`/offer`, `/accept`, `/decline`).
    - Peer discovery: mDNS is implemented for peer discovery, and also rendezvous servers can be set up without too much difficulty. Additionally, the manual `/ping <multiaddr>` command provides a mechanism that *can* work for cases in which mDNS fails.
    - Full documentation is found in `README.md`.
- Bonus Features:

    - Advanced UI using ratatui: This bonus feature has been implemented, providing a nice user experience with 3 panes, all with scrolling, multiple user input options and a graceful exit.
    - Multiple "Extra Features":
        - User list - Dynamic list of active users.
        - Directory safety - Manual setting of download directory validates paths.
        - Duplicate nickname handling - Allowed, but warns user when duplicate.
        - Visibility control - Users can toggle visibility using `/show` and `/hide`.
        - Self info command - Users can run `/me` to see info on themselves.
        - User info command - Users can run `/who <name>` to see info on another user.
        - Rendezvous servers - SwapBytes is compatible with a rendezvous server if this is a bonus feature (?).
        - Help command - Users can run `/help` to see all available commands.

5. Grade Estimate and Justification

Estimated Grade: A

Justification:

- The project implements all the specified basic requirements for the SwapBytes application. It features a functional P2P application for chat and file offering built on `libp2p` and Rust. Crucially, it goes beyond the basic requirements by implementing an advanced `ratatui` TUI and a few extra features listed above.
- The codebase is well-structured into modules with well commented code, and the `README.md` provides good documentation, instructions, demonstration, and discussion of potential issues and workarounds.
- Of course, I don't think this software is usable in the real world. It is insecure, and likely has many bugs. But, I had a lot of fun developing it to be something I think is neat, adding many subtle touches like message notifications, a real time user list and scrolling. I learnt a great deal in the process and I think that it demonstrates my creativity and my understanding of libp2p and Rust.
- The overall quality, completeness of basic features, inclusion of a bonus features, and thorough documentation justify an A grade.