Swarm Player - design / project plan

Milestone 1

Requirements

Implement auto-join procedure:

- The administrator configures public entry page: sets main app URL and hotspot name
- The user opens the public entry page in a browser
- The public entry page asks the user to join the private wifi hotspot
- The public entry page indicates when the user disconnects from the internet
- The public entry page indicates when the user reconnects to the internet (the user selected wrong hotspot, the server is down, etc.)
- The public entry page indicates when the user connects to the private hotspot
- Redirects to the main webapp
- The user is playing with the main webapp
- The main webapp indicates when the user disconnects from private hotspot
- The main webapp indicates when the user reconnects to the private hotspot (pobably only the server went down for a while)
- The main webapp indicates when the user connects to the internet
- The main webapp redirects to the public entry page

It's possible that the user's device connects both the internet and the private hotspot (has Ethernet and WiFi or 2x WiFi), in this case "re-connecting" to the internet can be very quick.

Hotspot switching should be emulated for development phase.

Server

Implement minimal functionality, just to support client auto-join feature:

- written in Rust
- clients can connect with websocket
- receive heartbeat or user input from client
- display client requests

Client

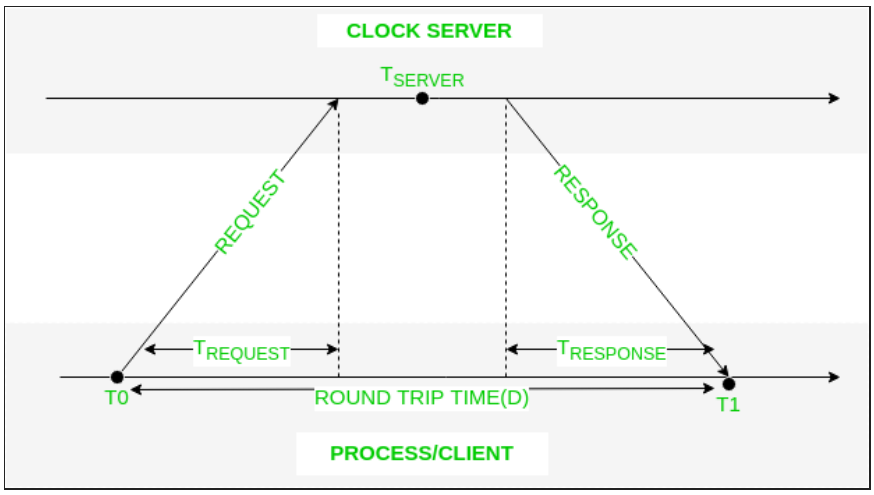
Implement auto-join:

- implement hotspot detection in public entry page
- the switch between entry page and main app should be "invisible"
- main webapp should have only minimal functionality
- implement hotspot switch detection in main webapp

Time sync

Was not planned in this milestone, but finally implemented for its simplicity.

The client uses Cristian's algorithm to synchronize time to the server.



The server only tells actual timestamp for the client's request.

Test features:

- the server simulates slow network, by sleeping the same amount before and after getting timestamp,
- the client's timestamp can be shifted to emulate clock skew.

Milestone 2

Requirement

Clients execute commands simultaneously, synchronized with each other.

The pre-requisite is met: time synchronization is already implemented.

Server

- The server receives message from external sources
- The server adds timestamp to the message: time of receipt plus official lag
- The official lag is cca. 50..100 ms: long enough for all clients to receive it in time, but short enough not to be able to hear the delay
- The server broadcasts all messages to all clients, no filtering applied in this milestone yet

Client

- Receives message, adds it to a container
- Retreives and processes item at the requested time
- Handle overdue messages
- Performs some simple visible action, e.g. sets background color