Systems Programming Project Report

Creators

Diogo Lee Leitão nº 99917 LEEC João Barreiros C. Rodrigues nº99968 LEEC

General Architecture

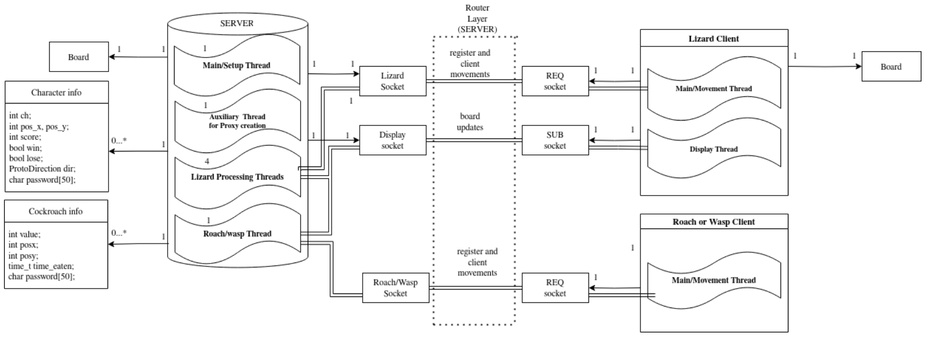

Figure 1: Simplified Architecture of the lizardsNroachesNwasps distributed system

Communication / Message Architecture

Two protobuffer messages which are sent in bitstream pack:

- **ProtoCharMessage**
  - Used to register and play the game, contains information about the character(s) movement(s)
    - Type 0: Lizard register
      - The reply to this message is the char the lizard will assume (for example if the char was already taken or if the server is full)
    - Type 1: Lizard movement
      - The reply is a dud (dummy) since we all REQ messages must receive a REP message.
    - Type 2: Display register
      - The reply is a hard-coded C-struct aka board
    - Type 3: Cockroach register
      - The reply has the character zero in the ch field if the server is full
    - Type 4: Cockroach Movement
      - Dummy reply
    - Type 5: Lizard disconnect
      - Dummy reply
    - Type 6: Wasp register

        - The reply has the character zero in the ch field if the server is full
      - Type 7: Wasp Movement
        - Dummy reply
      - Type 8: Cockroach disconnect
        - Dummy reply
      - Type 9: Wasp disconnect
        - Dummy reply
- **ProtoDisplayMessage**
  - Used to update remote displays in lizard clients

One C structure that is hard-code sent

- **Board**
  - Since the lizards do not need to communicate exclusively in protobuffer Messages a C structure is sent hardcoded through the ZMQ-REP server socket after the lizard client regists its display in the server so that the lizard's display can be initialized with the current state of the game.
  - A containerized version of the struct could have been done in protobuffer, but we opted to not change this as it was not required.

Implemented Functionalities

General/Meta

[x] Simple Message Authentication using passwords

Server

[X] Threaded Server with 4 threads for Lizard Handling and 1 for Roach/Wasp Handling

[X] Include win and lose conditions

Lizard-Client

[X] Seamlessly integrate remote display

[X] Timeout Inactive Clients

[X] Handle SIGNINT exit

Roach/Wasp client

[X] Handle SIGINT exit

[X] Create client in a non-C language

Major alterations between versions

In order to support heterogenity both ZMQ send and receive were changed. Now we pack the protobuffer into a bitstream to be sent and the message reception is done with the help of a zmq_msg_t variable to support variable sizes.

The lizard client was threaded to support a keyboard controller and a remote display.

Has mencioned above the server has also been threaded to support multiple lizard and roach message handling