Network Protocol for Online Super Tetris

ECE 462

Purdue University, West Lafayette, IN

October 9, 2009

Objective

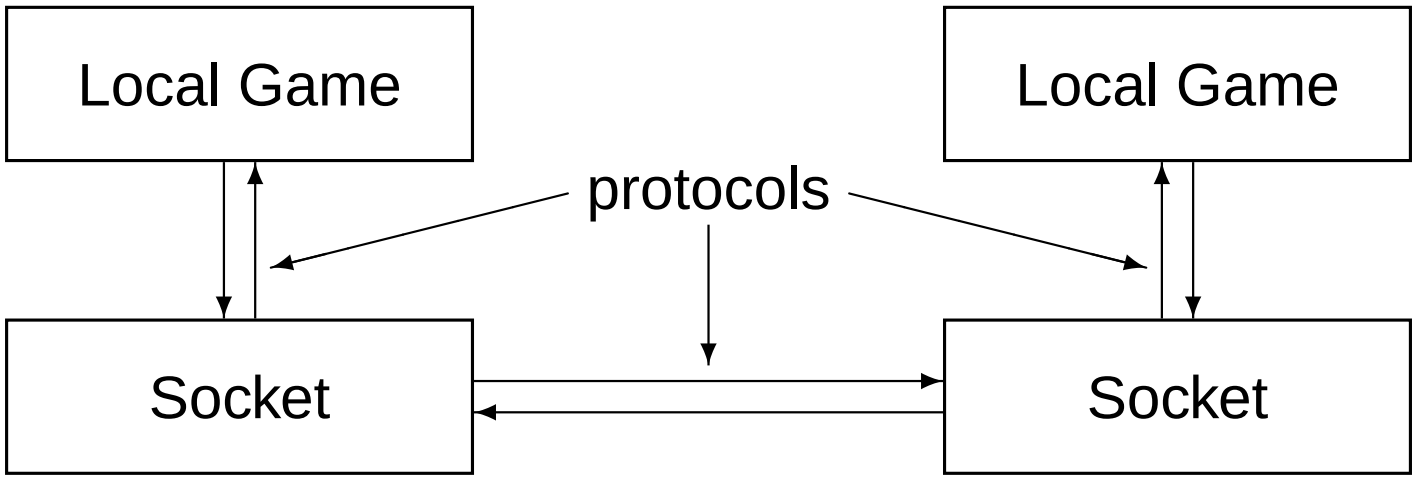
We need a network protocol because...

- people talk only when they speak the *same* language
- make design work easier for both server and client sides

What is Protocol?

...“a set of instructions for transferring data.”

—*Wikipedia*



General Rules

- All operations must be acknowledged by server.
- Client must not act on its own.

$\Rightarrow$ Client relays player actions to server.

- Server decides whether or not to take the actions, and send the action back to client if permitted.
- Client acts only upon receiving an action from server.

General Rules

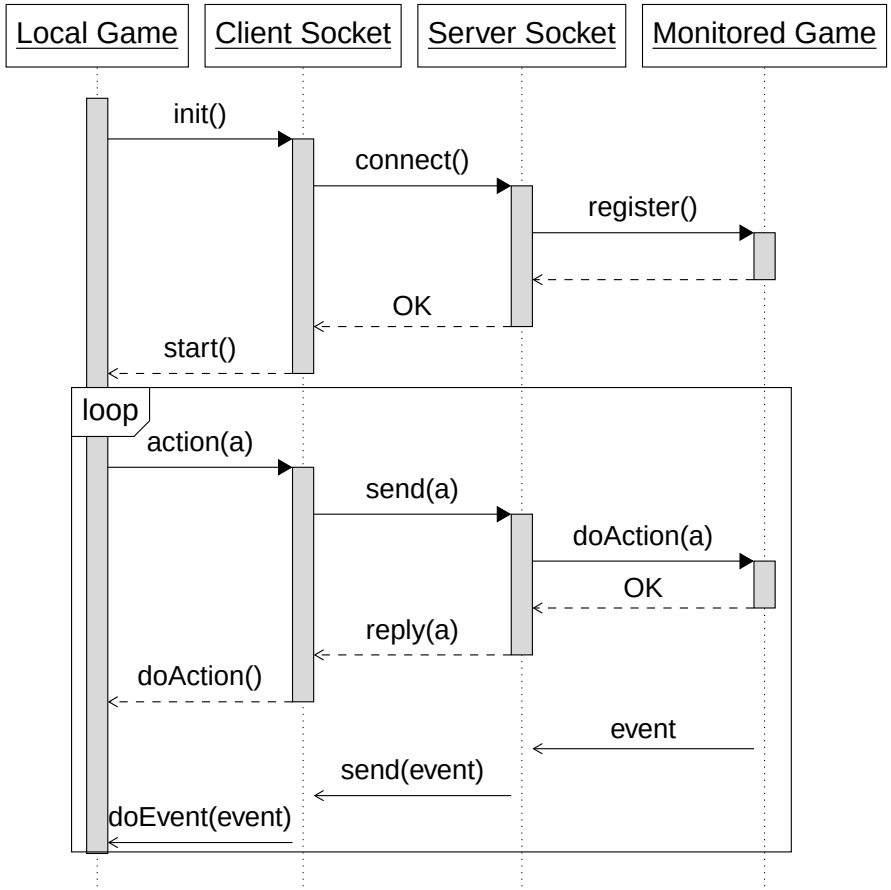


Figure: UML Sequence Diagram

General Message Format

- `<COMMAND>?`: ask for client information
- `<COMMAND>␣[<argument>]`: control messages (player join, quit, start, etc.)
- `<id>:<COMMAND>`: action triggered by player <*id*>, or a control message particularly for player <*id*> (e.g. fall, left, gameover...)

Establishing Connection

- Client: `connect(addr, port)`
- Server: `accept()`; `"ID?"`
- Client: "`ID=<id>`" (`id`: a unique identifier)
- Server: `accept()`; `"GAMETYPE?"`
- Client: "`GAMETYPE=tetris`"
- Server: `"ACCEPTED."` (if `id` is unique and `gametype` is recognizeable.

Initializing Game

- Server: “READY?”
- Client: “READY”

- Server: “JOIN <id>” (when rival arrives)
- Server: “QUIT <id>” (when rival leaves)

Running Game

Preparation:

- Server: “<id>:PIECE <piece>”
- Server: “START”

In game:

- Server: “<id>:<COMMAND> [<argument>]”, <COMMAND> can be:
  - PIECE <piece>: a piece queued to appear
  - LEFT, RIGHT, ROTATE, FALL: Tetris operation
  - ATTACK <line>: A line transferred to this player
  - GAMEOVER: Game over detected
  - WIN, LOSE: Final result of the game

How to Respond

- `PIECE <piece>`: a piece queued to appear
  - Put the piece into the “next piece” queue
- `LEFT,RIGHT,ROTATE,FALL`: Tetris operation
  - Perform the corresponding action
- `ATTACK <line>`: A line transferred to this player
  - Put the line at the bottom of the playfield
- `GAMEOVER`: Game over detected
  - Stop the game
- `WIN,LOSE`: Final result of the game
  - Display the result (e.g. message dialog) and reset the game

Client Actions

Whenever the *local* player triggers an action, for example, pressed the *Left* key:

- send “`<id>:LEFT`” to server, where `id` is the ID string of the local player and... do nothing!

Conclusion

1. Connect to server
2. Shake hands (`ID, GAMETYPE, READY`)
3. Perform only actions from server
4. Relay local actions to server (and wait for feedback)

Questions?