Flow Chart for Gobang

1. The Overview Model

The overview model for Gobang:

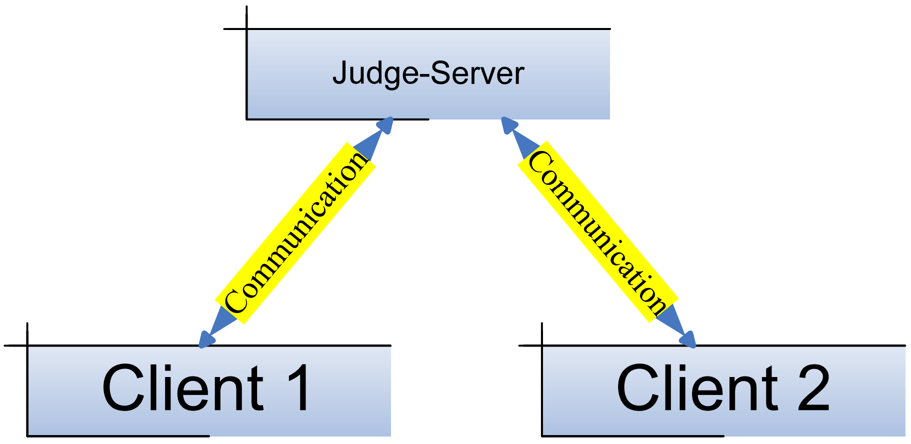

chart 1 Overview Model for Gobang

2. Client Model

The Client Finite State Machine for Gobang:

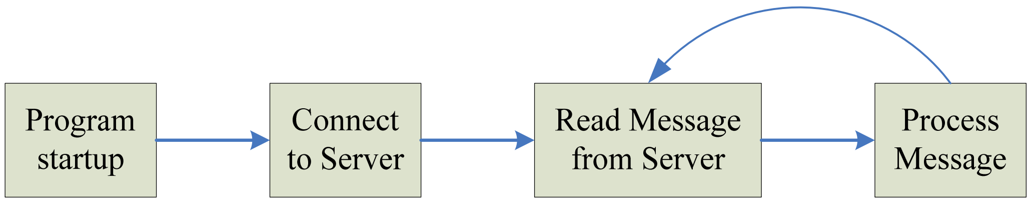

chart 2 Client Finite State Machine for Gobang

We advise the client should keep reading the message infinitely from the socket and then process the message.

For every message that client send to server, the client will first get the response message with the value COMM_MSG_ACCEPTED or COMM_MSG_REJECTED from server. The chessboard state will be changed only when the server sends COMM_MSG_CHESS to both clients, followed with a data sequence of struct chess_info. So the client is not allowed to change the chessboard state when it generates a chess step, just send it to the server please (so client will receive its own generated chess information if the former COMM_MSG_CHESS is accepted by the server).

When the client has connected to server, the detail Finite State Machine for the processing is shown bellow.

Client may receive COMM_MSG_GAME_START, COMM_MSG_FIRST, COMM_MSG_SECOND before game start. Also, it can send COMM_MSG_GAME_REQUIRE_START, COMM_GAME_FIRST and COMM_GAME_SECOND to server at any time before game start.

States before game start:

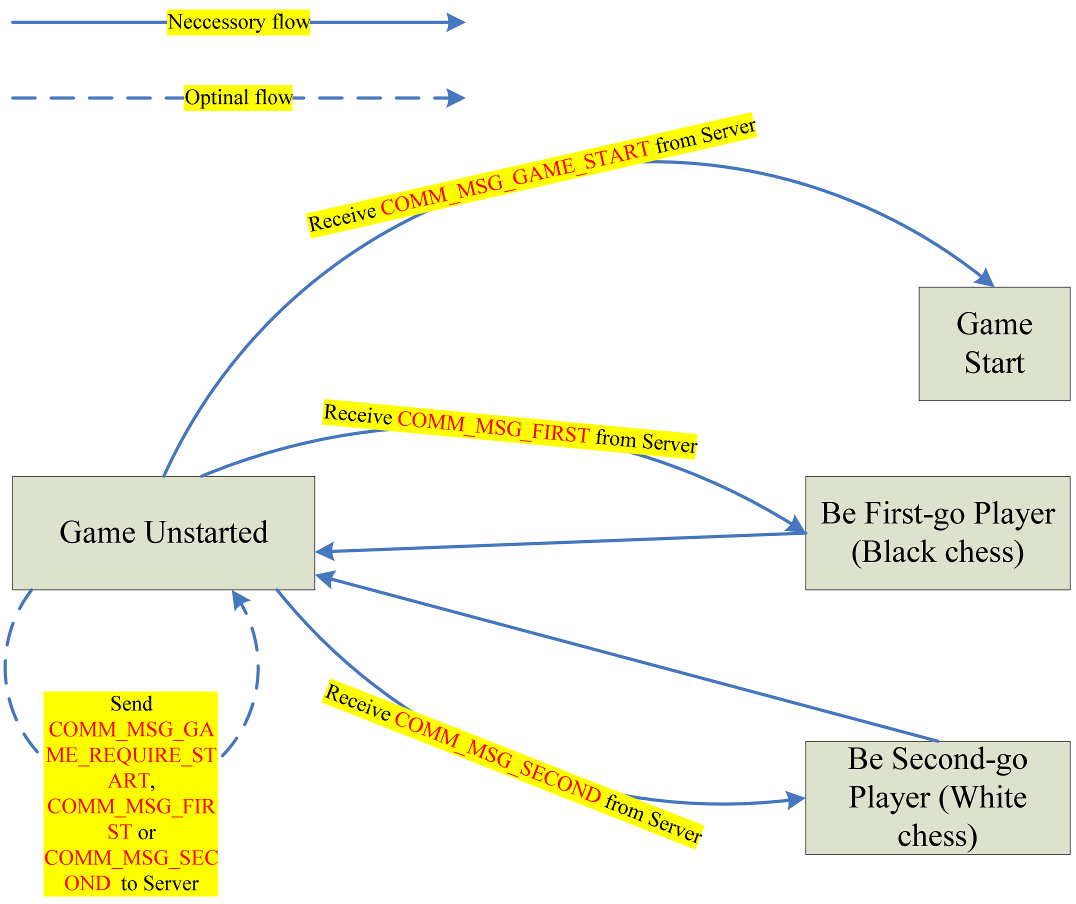
Neccessory flow
Optinal flow
Receive COMM_MSG_GAME_START from Server
Game Start
Receive COMM_MSG_FIRST from Server
Game Unstarted
Be First-go Player (Black chess)
Send COMM_MSG_GAME_REQUIRE_START, COMM_MSG_FIRST or COMM_MSG_SECOND to Server
Receive COMM_MSG_SECOND from Server
Be Second-go Player (White chess)

chart 3 State Machine for client before game start

States when game is playing:

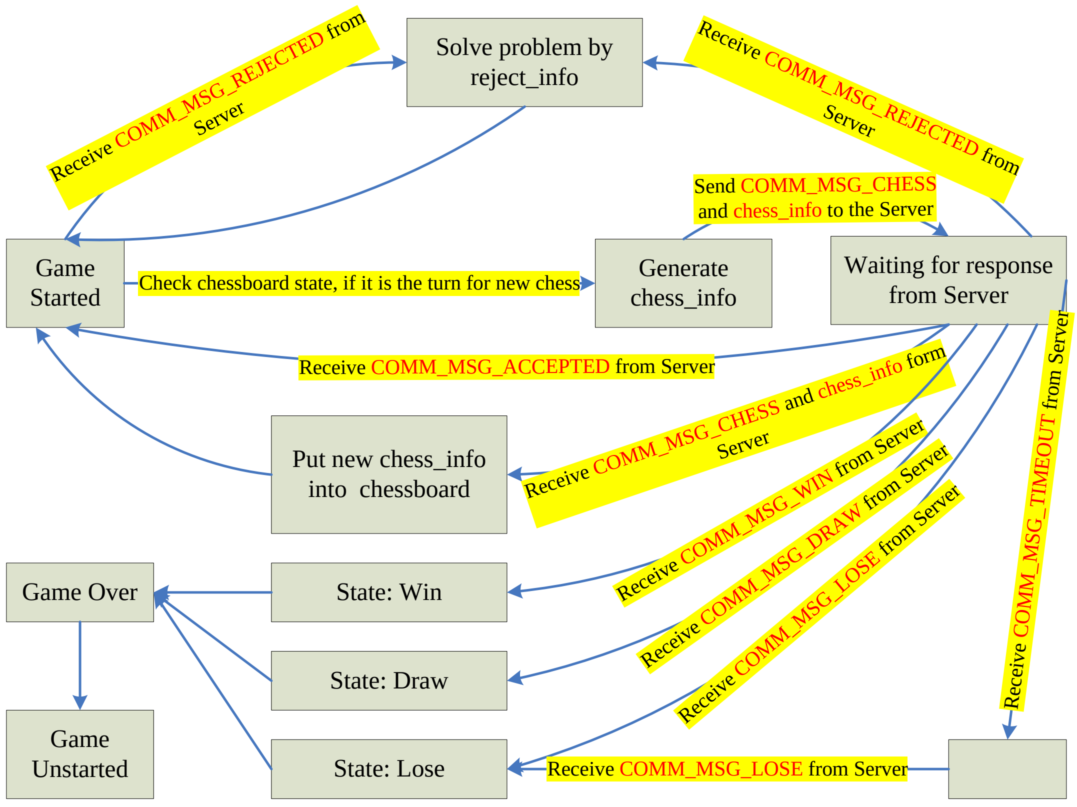
Solve problem by reject_info
Receive COMM_MSG_REJECTED from Server
Receive COMM_MSG_REJECTED from Server
Send COMM_MSG_CHESS and chess_info to the Server
Game Started
Check chessboard state, if it is the turn for new chess
Generate chess_info
Waiting for response from Server
Receive COMM_MSG_ACCEPTED from Server
Put new chess_info into chessboard
Receive COMM_MSG_CHESS and chess_info form Server
Receive COMM_MSG_TIMEOUT from Server
Game Over
State: Win
Receive COMM_MSG_WIN from Server
State: Draw
Receive COMM_MSG_DRAW from Server
Game Unstarted
State: Lose
Receive COMM_MSG_LOSE from Server
Receive COMM_MSG_LOSE from Server

chart 4 State Machine for client when game is playing

To decide whether it is time for the client to generate new chess information and send the chess information to the server, just check the state of the current chessboard and the chess color of this client (Black chess for First-go player and White for Second-go player).

3. Server Model

State machine before game start:

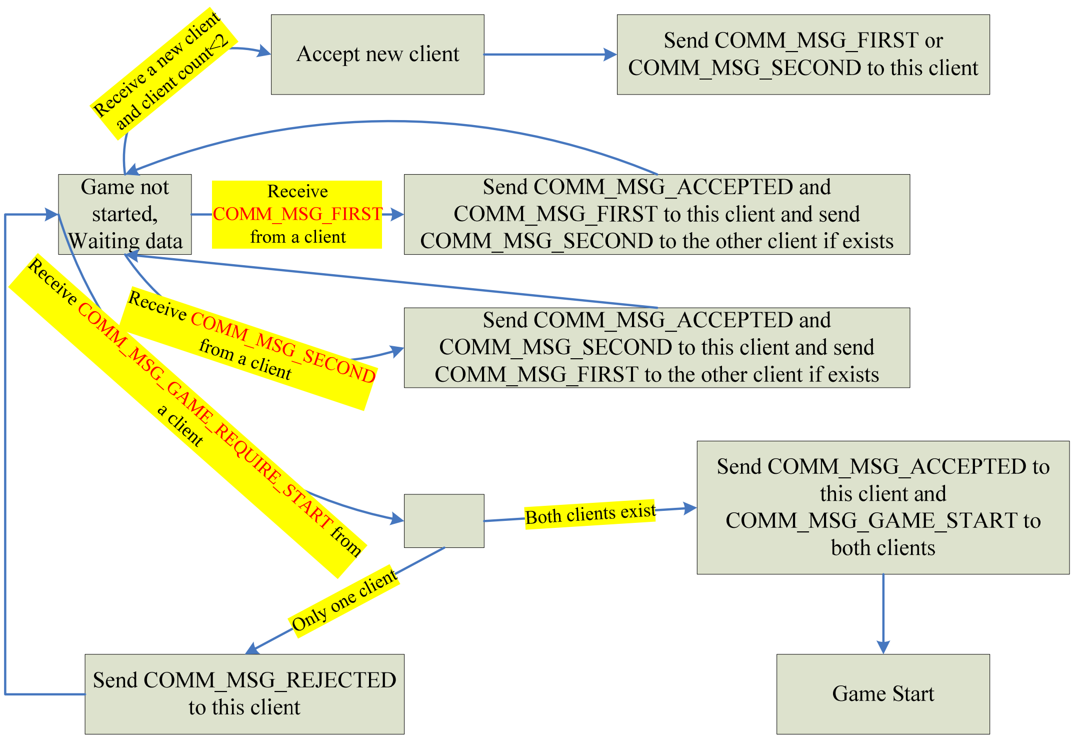
Receive a new client and client count<2
Accept new client
Send COMM_MSG_FIRST or COMM_MSG_SECOND to this client
Game not started, Waiting data
Receive COMM_MSG_FIRST from a client
Send COMM_MSG_ACCEPTED and COMM_MSG_FIRST to this client and send COMM_MSG_SECOND to the other client if exists
Receive COMM_MSG_SECOND from a client
Send COMM_MSG_ACCEPTED and COMM_MSG_SECOND to this client and send COMM_MSG_FIRST to the other client if exists
Receive COMM_MSG_GAME_REQUIRE_START from a client
Both clients exist
Send COMM_MSG_ACCEPTED to this client and COMM_MSG_GAME_START to both clients
Only one client
Send COMM_MSG_REJECTED to this client
Game Start

chart 5 Server State Machine before game start

State machine when game is playing:

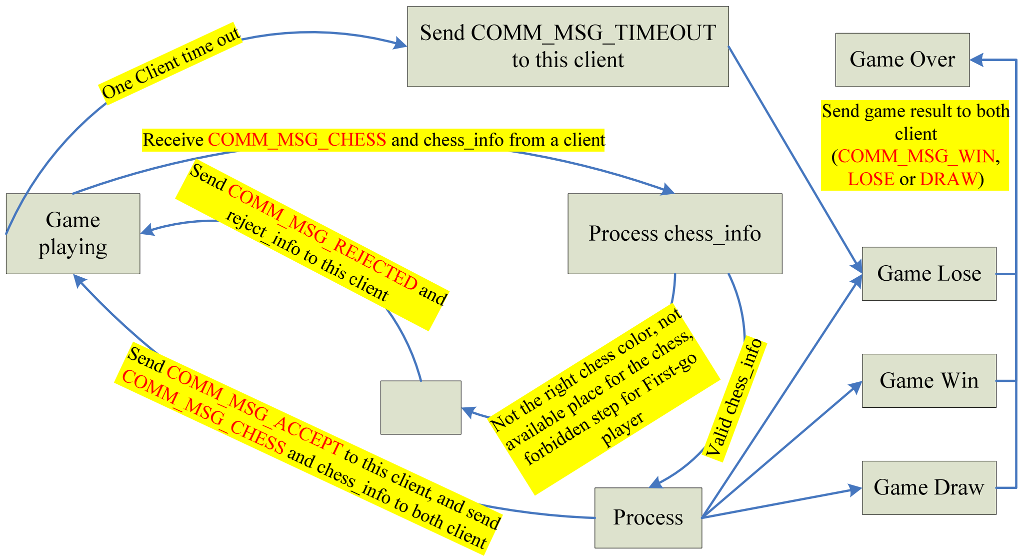
One Client time out
Send COMM_MSG_TIMEOUT to this client
Game Over
Send game result to both client (COMM_MSG_WIN, LOSE or DRAW)
Receive COMM_MSG_CHESS and chess_info from a client
Game playing
Send COMM_MSG_REJECTED and reject_info to this client
Process chess_info
Game Lose
Not the right chess color, not available place for the chess, forbidden step for First-go player
Valid chess_info
Game Win
Send COMM_MSG_ACCEPT to this client, and send COMM_MSG_CHESS and chess_info to both client
Process
Game Draw

chart 6 server State Machine when game is playing