Homework 2

Hadachi&Lind

November 29, 2017

Must Read:

1. Deadline for doing homework is 5 weeks starting from now "2017.11.29"

   due date is: **2018.01.05 5:59:59 EET**
2. For any delay in submitting the homework they be will a penalty (-5% of the grade for every day delay)
3. You can work in groups of maximum 4 people
4. Python 2.7 or 3.x only
5. You should create a public git repository for your project and share it.
6. Be aware that the git should contain a time line progress of your implementation in case there is only one big commit at the end you will be penalized
7. You are allowed to use any fancy open source library as long as you fulfill the requirements.
8. In case of using third party libraries (not in Python standard distribution) you should reference them in your documentation (be aware of the licensing of the dependencies - we only accept those of open-source kind). Be aware that in case you use any ready product or source code snippet of any existing application unreferenced - it will be considered as plagiarism.

1 Introduction

The main objective of this homework is to demonstrate how to handle Remote Procedure Calls, Remote Objects, Indirect Communication and Peer-to-Peer technologies. Therefore, we will modify our existing Sudoku-Multiplayer game implementation. Here we suggest several levels of upgrading it:

1. (Obligatory) Removing the Custom written application protocol (multi-player network protocol you designed to make the game instance accessible over the network to the game clients). Employing RPC or Remote Objects paradigms instead of custom application protocol on TCP/UDP. Choice of RPC or Remote Objects approaches are up to you (both cases are suitable).
2. (Obligatory) Automatic service discover by game clients. Just like in Quake multi-player setup - the game server(s) has/have to be automatically discoverable by clients without a need to enter the server address manually. Can be implemented for local scope (one private subnet LAN - using UDP broadcast or several private networks - using UDP multi-cast). More complex scenarios for global scope are explain below.
3. (Optional) Introducing the aspects of indirect communication. In this sense even the RPC and Remote Objects assume direct communication (TCP/UDP) between client and server. Following the indirect communication - the client should be decoupled in time and space (follow seminar6 document). The objective here is to select the most suitable middleware

providing indirect communication (message queues, message brokers, group communication) and implement the communication of Sudoku-Solver's client and server on top of the chosen middleware (having RPC or Remote Objects here is the matter of preferred indirect communication middleware as many of them already suggest RPC or Remote Objects API).

4. (Optional) Service discovery on global scale (not just one LAN or one ISP, but several LANs in several ISPs) using Peer-to-Peer technologies. Clients must be able to discover the game server even if the server resides in different LAN in different ISP private subnet (not visible directly for incoming connections). The goal here is to pick up the Peer-to-Peer design (structured-centralized, structured-decentralized, flat-centralized, flat-decentralized, hybrid, own-designed ... etc. follow p2p lecture slides and p2p seminar) and employ it to provide service discovery for the Sudoku Solver's client and server. Server must be able to announce his presence to Peer-to-Peer framework, Clients must be able to lookup for games servers using Peer-to-Peer framework. Applicable for both direct- and indirect communication paradigms. In case of skipping the indirect communication tasks - your Peer-to-Peer framework must suggest "mediation" for helping client and server establish direct connection or provide "overlay" in case direct connection between client and server is not possible. In case the direct communication is implemented - the task of the Peer-to-Peer framework is rather overhead! it can be reduced to service discovery (for example suggesting nearest message broker for client or server). You can design you own simple Peer-to-Peer framework or rely on existing libraries (see p2p seminar).

1.1 Concurrent Sudoku Solver

(This section is the same to the one from HW1)

2 General Network Architecture

The general architecture is illustrated in figure 1 and it is structured as follows:
Dedicated Game server is responsible for:

- in case of using direct communication and no P2P: announcing server's presence into LAN (broadcast/multi-cast)
- in case of using direct communication and P2P: announcing server's presence Peer-to-Peer framework
- in case of using indirect communication and no P2P: registering server into Indirect Communication Middleware (contacting the message broker, or joining the group communication etc.)
- in case of using indirect communication and P2P: ... up to you choice
- creating game sessions on demand and generating Sudoku to be solved
- handling game sessions and corresponding players (having a player associated to corresponding connections is up to you and is dependent on your scenarios choice - whether you do RPC or Remote Objects and whether you do indirect-communication scenario).
- counting players activities giving the scores for correct guesses or reducing scores for bad guesses.

Game client is responsible for:

- in case of using direct communication: looking up the list of active servers (dependent on your scenarios choice - on the LAN scope using broadcast/multi-cast, or on the global scope using Peer-to-Peer framework)
- in case of using indirect communication: in addition to listening for multicast/broadcast, client asks message broker for active game servers from indirect communication middleware (subscribing for a message channel on broker).

- handling user input
- showing dialog: provide player nickname
- showing dialog: provide server address
- showing dialog: chose to join existing session or to start the new one
- rendering current game state (Sudoku)
- rendering score of the players

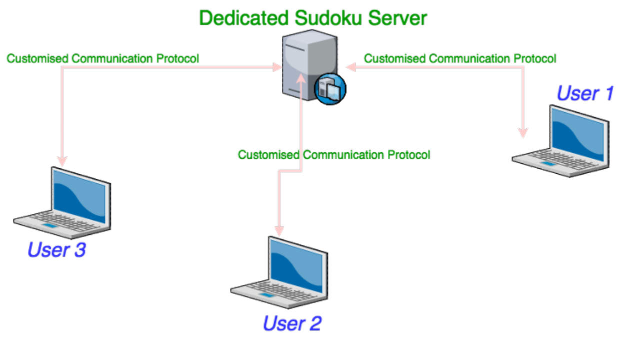
Dedicated Sudoku Server
Customised Communication Protocol
Customised Communication Protocol
User 1
Customised Communication Protocol
User 3
User 2

Figure 1: Overview of the general architecture. Here "Customized Communication protocol is your-suggested application layer protocol on top of TCP or UDP

3 Requirements

The requirements remain the as those in HW1 - keeping in mind several aspects:

1. try to avoid using socket API when possible (doing broadcast/multi-cast still assumes you use socket API)
2. respect the chosen network scenarios (RPC or RemoteObjects, direct or indirect, simple or P2P service discovery).

4 Implementation

You are free to use any third party libraries for GUI, user input processing etc. including for network handling as long as you stay on Python 2.7 (3.x) and fulfill the requirements. You can choose any GUI design as long as the game requirements are respected. The use of threading is up to you and you can avoided them as long as other requirements are preserved.

5 Deliverables

They will be two major deliverable for this Homework:
First: You have to submit your Team members names including you and also the public git repository link via the Moodle, latest 13st December at 15h00.
Moodle Link
Second: A report that includes the following:

- Architectural document illustrating how RPC or Remote Objects were integrated, how Indirect Communication paradigm was employed (if it was), how the service discovery is organized. In case Peer-to-Peer framework was in use: explain it's design (if custom one is made), in case third party framework was used - explain how the service discovery is performed.
- User manual explaining setup process: Middleware(if used), Peer-to-Peer framework (if used), Client, Server, and (other possible dependencies required).

- User manual explaining the user interface and how to play the game.

Third: The source code of your implementation.
Concerning the implementation, we will consider only the source code from your public git repo provided. Note, any modification done after the due date to the git won't be considered. Concerning the report, it should be PDF document and submitted in the course website by one member of the team.
Submit Link

6 Grading

Total points for this Homework is 20 pts, plus 10 bonus pts and its breakdown is as follows:
• 6 pts: The user manual document
• 24 pts: The implementation:[
– 2pts: code is functional (and does run)
– 6pts: all obligatory tasks are done and are functional
– 4pts: requirements were respected
– 2pts: code is documented]
• 10pts: (Bonus) one optional task is done, is functional and is documented.