VirginiaTech
Invent the Future

Assignment 4

Cooperative Multiplayer Minecraft

Due Date: Monday, April 10, 2017 @ 23:55

Credit to Akshay Shastry and Prakriti Gupta

ECE 4564 - Network Application Design

ece
BRADLEY DEPARTMENT
of ELECTRICAL & COMPUTER ENGINEERING

Learning Objectives

COAP
- GET requests
- POST requests

Minecraft API

Encapsulation/Decapsulation

Raspberry Pi GPIO

Data Serialization - Pickling

Assignment Overview

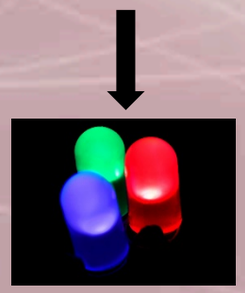
Assignment 4 is on a multiplayer minecraft game. The task is to Cooperatively build a wall.

The system uses four Rpi's following the client/server model discussed in class. The server runs the minecraft game and talks to clients who are players

The client Rpis are players each with a different color. Player Rpi "gets" the current position of the pointer in the game and "posts" a new command to build a block on the current instance of the game.

System Overview

MineServer



LED indicating Token

GET (Position, Token) , iff Token == valid, then
POST (place Block at POS)

Client A -- Red

GET (Position, Token) , iff Token == valid, then
POST (place Block at POS)

Client B -- Blue

GET (Position, Token) , iff Token == valid, then
POST (place Block at POS)

Client C -- Green

Client/Server Model

- **Client RPI**
  - Builds and sends GET request to server Rpi via COAP to get current position of the builder
  - Receives and prepares a request to POST to server in order to build a block at a particular position in Minecraft game, again via COAP
- **Server RPI**
  - Receives request payload from client RPIs, requesting position of the builder, and the Token which specifies the active player/client
  - Sends position information (X, Y, Z) to all client RPIs responding to GET()
  - Receives POST from active player/client to place the wall at position (X + 1, Y, Z)
  - Interact with the Minecraft API to command it to build the block in the specified/current location
  - Update builder position, Update Token of next player
  - Indicates current client's colorID using RGB LED via GPIO pins

Client Requirements

- Client RPI
  - “GET” current position of the minecraft game pointer from the Minecraft Rpi using COAP
  - Unpickle the position and identify next position to build a wall, pickle it
  - “POST” the pickled position to Minecraft RPI using COAP
  - Waits for it’s next turn and repeat the same

- Server RPI
  - Receive “GET” request via COAP
  - Get current location information from Minecraft using Minecraft API
  - Pickle the position and send it to client Rpi using COAP
  - Receive “POST” request from client using COAP
  - Unpickle it, and send a command to Mincecraft to place a block at the specified location

Build A Wall!

- The ultimate task is to build a wall, 10 spaces wide in X – axis, 2 spaces high in Y – axis
- Each Client must have its own separate block type (Wood, diamond, stone), format it using block.id or the number system
- Each Client gets to place only one block at a time
- The server uses a round robin token mechanism, indicating the active player, this needs to be implemented by you as well
- If you feel this is very easy, you are welcome to build a pyramid! No extra points though!

Client Command Line Parameters

`mineclient.py “Server_ip_address”`

Example:

python3 ./mineclient.py “172.48.212.22”

Game Steps

- Server starts the game (You can use GUI) and waits for GET request payload from players
- Server Receives and unpickles GET request, replies to player with current position (Character's position initially) from Minecraft API after pickling the position data and the Token
- Clients receive position and Token, if Token is valid, i.e. Client A is indicated by 1 and so on,
- The Active Client sends a PUT request with the position of the block and ClientID
- Server receives and unpickle PUT request, adds a block to minecraft game using commands from mincecraft API
- Server Updates Token i.e. Token = player B(You can use numbers as well, A=1, B=2, C=3) and position of the recent block placed
- Server-Client interaction Repeats the steps till the wall completes
- Server has to notify the clients that the game is completed when the wall completes

Payloads

GET Response Payload (Python tuple):

- X coordiante
- Y coordinate
- Z coordinate
- Token

PUT Request Payload (Python tuple):

- Player_Id
- X coordiante
- Y coordinate
- Z coordinate
- Block Type

AIOCOAP Library

- https://github.com/chrysn/aiocoap

This library has to be installed on Python 3.5.x only.
Make sure you have Python 3.5.x installed

Minecraft API

https://github.com/py3minepi/py3minepi

Library to be installed on Python 3.5

Python 3.5.x

Steps:

sudo apt-get update
sudo apt-get upgrade
wget https://www.python.org/ftp/python/3.5.1/Python-3.5.1.tgz
tar -zxvf Python-3.5.1.tgz
cd Python-3.5.1
./configure && make && sudo make install

Check for python version then …..

sudo pip3 install --upgrade pip
sudo pip3 install --upgrade "aiocoap[all]"

Approximately takes 30 mins! Also you might face PIP3 issues!
So Please start early! Alternatively for python 3.5.2 steps are posted as well

The Grading and Validation Process

We expect you to run the server and the clients just once, and manage to complete the task

MineServer – 50 points

- Server should run continuously, clients should run as long as the game is in progress and then exit – 5 points
- Server notify that the wall is complete, in the GUI of the Minecraft Game as well as to the clients in a RESTful interaction (You can slightly cheat here by denoting a number 0 on the token indicating all tasks complete, and transmitting as part of GET) – 5 points
- REST interactions using CoAP – 20 points
  - GET – 10 points
  - POST – 10 points
- Pickling data – 5 points
- Token implementation – 10 points
- LED indication for player – 5 points

The Grading and Validation Process

We expect you to run the server and the clients just once, and manage to complete the task

Clients – 40 points

- RESTful interactions using CoAP – 20 points
  - GET – 10 points
  - POST – 10 points
- All 3 Clients build wall turn by turn – 10 points
- Pickling data – 10 points

Report – 10 points

Report

You must document the design, and outcomes in a brief written report (2 pages). One report submitted by each team. Your report should contain the following items.

- At the top of the first page of your report, include: your names (as recorded by the university); your email address; and the assignment name (e.g., “ECE 4564, Assignment 1”). Do not include your Virginia Tech ID number or your social security number.

- The body of the report must contain the following sections. Use section numbers and headings to organize your report.
  - Section 1 – Objectives: Provide a description of the design objectives and general approach to the design.
  - Section 2 – Team member responsibilities
  - Section 3 – Conclusions: Discuss the outcome of your design and any problems encountered and resolutions; what you learned by doing this project; and any experiences that were particularly good or bad.

Python Style

Follow style guide PEP0008 when writing and commenting your code

https://www.python.org/dev/peps/pep-0008/

Coding for all assignments in Python 3

What You Turn In

All assignments must be submitted through Canvas, no later than the due date of 2017 April 10 @ 23:55

Your assignment should be a single zip or tarball (i.e. tar.gz, tar.bz) which contains the following:

- All source code for this assignment
  - Python code running on client and server Rpi's
    - Client codes identified as "clientA.py", "clientB.py", "clientC.py",
    - Server code identified as "MineServer.py"
- A ReadMe file describing client and server initialization procedures and any extra libraries used.
- Report (PDF file)

Assignment References

The official WolframAlpha API

- http://products.wolframalpha.com/api/

“Beej's Guide to Network Programming Using Internet Sockets” (PDF)

“Foundations of Python Network Programming”, 2nd Ed. (Full text – VT Library)

Python Network Programming

- http://ilab.cs.byu.edu/python/

Assignment References

Raspberry Pi – GPIO References

Raspberry Pinout

Rpi Low-level Peripherals

Rpi.GPIO

Academic Integrity

- For this assignment, it is expected that a team's work is their own
- The code you turn in must be your own (i.e. you need to have written your assignment)
- You are allowed to copy and paste example code from other websites, but you must include a comment in your code that attributes the website you copied the code from (i.e. original author's name and URL to the original code)
- You can discuss the assignment with other teams
- However, you cannot just tell another team the answer to a particular problem

Final Thoughts

In many cases, engineers are expected to just make things work given a particular design constraint (e.g. software package to use or are limited to a particular hardware platform).

You will likely run into similar situations in this class while designing and implementing your assignments and project.

.

When you're stuck, try searching online for a solution. Many times others have tried something similar and documented their experiences for others to learn and benefit from

Do not publically post answers to assignments, or your code until after the assignment due date.

Contact your instructor or GTA's as soon as you encounter a problem you're unable to solve. Don't wait to begin right before the assignment is due.