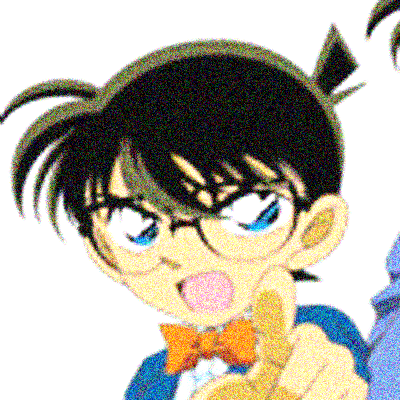
**EC4206 Networking Project (원안)**

***Who was the Greediest last week?***

by 20105043 손선일, 20165142 이정호

**GIST**

**EC4206 Networking Project (제목수정)**



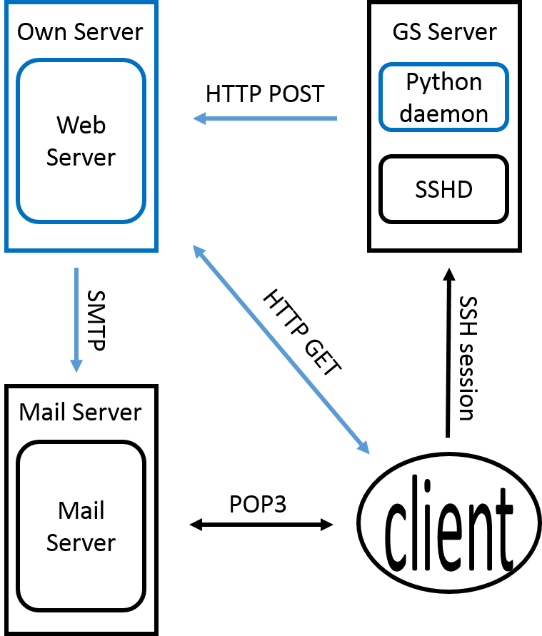
***DetectiveConan.py***

***codename: Newton's Server***

by 20105043 손선일, 20165142 이정호

**GIST**

**How does it work? At a glance**



Conan.py (python daemon) will inspect each user on newton server according to their CPU usage. And it would reported by json DB to be published as html and posted on web-server.

**Python modules for constructing the project**

USE python ‘**subprocess**’ module to invoke external processes.

USE python ‘**json**’ module for json decode/encode/manipulate.

USE python ‘**smtplib**’ module for sending mail.

USE python ‘**simpleHTTPServer**’ module for web server implementation.

**Components & Details**

**—modules and functions to be implemented**

*Functions here are described as…*

return\_type func\_name()

*(if there is no return value, return\_type is not specified)*

**1. Python Daemon (Conan.py)—A niced process: running all the time on the server but should yield resources properly when required.**

**Json invokeTop()**

# Invoke ‘top’ process

# parse result of ‘top’ into json format

**Json invokeWho()**

# Invoke ‘who’ process

# parse result of ‘who’ into json format

**Json mergeJson(whoJson, topJson)**

# merge two json files which contain parsed result of ‘who’ and ‘top’

for user in whoJson.users:

if user is in topJson.users list: continue

elif user is only in the whoJson: add user with default null fields

else: delete user (system users are ignored)

**checkHeavyUser(Json)**

# if an excessive user (monopolizing cpu resource) exists

# then `write` warning message to the user’s terminal directly.

**saveJson(Json)**

# append Json data to maintained file.

**sendJson()**

# report maintained file to our own server

# using HTTP POST method

# once per week (reporting interval surely adjustable)

**examineSystem() : main() like function for the daemon**

# examine system by

# invoking system processes ( **invokeTop() & invokeWho()**)

# analyzing and logging data ( **mergeJson(whoJson, topJson)** & **checkHeavyUser(Json)** )

# then save and (or) send data to server ( **saveJson(Json) & sendJson(Json)** )

# called regularly (once per 5 min)

**2. Web Server**

**2.A. Daemon listener:**

Listening to HTTP POST request from python daemon on GS server. On request, server works as follows.

**updateData()**

# update data with received json file

# load up new json data into memory

**analyzeData(Json)**

# sort users according to three legend.

# Using time, max cpu usage, avg cpu usage

**generateHTML(Analysis)**

# generate dashboard page from analysis.

# barely containing title and tables

**sendMail(Analysis)**

# send mail of congratulations to the top 3 resource bruisers (for each scoring standard in **analyzeData()**)

# this surely needs users’ email address, but we assume they need to be registered with email when opening accounts.

**2.B.Client listener:**

Listening to HTTP GET request for dashboard page. HTML file already provided by daemon listener, just send the file to client.

**Appendix**

**JSON format:**

Object {

user:string,

cpu:float,

process\_name:string,

timestamp:string

}

**Dashboard format:**

