Smart Farm 애플리케이션을 통해 분석한 Flask HTTP 성능

윤보민, 최완우

경희대학교

jellytrain@khu.ac.kr, htc.kovsky@gmail.com

Flask HTTP performance analyzed via Smart Farm application

Yoon Bomin, Choi Wan Woo

Kyunghee Univ.

요 약

본 논문에서는 Flask로 만든 HTTP 서버의 네트워크 성능을 Smart Farm 애플리케이션을 활용해 분석한다. 서버로는 Raspberry Pi 3B+ 모델을 사용했으며 Flask 모듈로 HTTP 서버를, PyMySQL 모듈로 데이터베이스를 구성했다. 개발 언어는 Python이다. 클라이언트인 Smart Farm 애플리케이션은 Unity를 통해 구현했다. 기본 토대는 Unity 엔진의 에디터로, 스크립팅 언어는 C#이다. 서버와 클라이언트를 구현한 후 더미 데이터를 통해 실험을 진행했다. 분석 결과, 데이터를 자주 주고받지 않아도 되는 애플리케이션에서는 주 데이터 교환 방식으로 HTTP 통신을 선택하는 것이 유리함을 확인할 수 있었다.

Ⅰ. 서 론

처음에는 Unity 기반 3D 애플리케이션으로 Smart Farm 기술을 활용한 가상 농장 경영 게임을 만들 계획이었다. 일본의 대형 유통기업인 라쿠텐과 벤처기업 텔레팜이 공동으로 개발한 '라그리'와 비슷하다. (1) 그러나 농사 대행에 가까운 이 애플리케이션과는 달리 게임과 같은 형태로, 소비자가 Smart Farm 기술을 바탕으로 가상 농장을 운영하는 느낌을 주는 것을 목표로 했다.

이를 구현하는 개발 도구로는 Python 기반의 Raspberry Pi와 에디터, C# 기반의 Unity를 선택했다. Raspberry Pi 서버와 Unity 클라이언트의 통신 방법으로 Socket 통신을 고려하다가 HTTP를 활용하는 것이 더 편리할 것으로 판단했다. 하지만 애플리케이션에 추가할 수 있는 기능을 구상하면서 HTTP를 어디까지 활용할 수 있을지 의문을 갖게 되었다. HTTP를 기반으로 통신할 경우 작은 데이터를 여러 번 보내는 것이 효율적이지 않음은 일반적으로 잘 알려져 있다. 우리는 가상 농장 경영 게임 플랫폼을 부분 구현한 후, 이를 통해 HTTP의 한계와 성능을 확인함으로써 HTTP를 사용해도 무방한 수준을 알아보고자 한다.

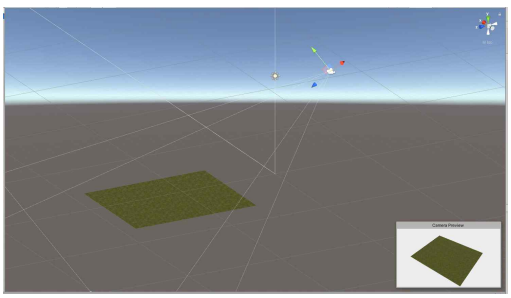
Ⅱ. 본론

1. 설계

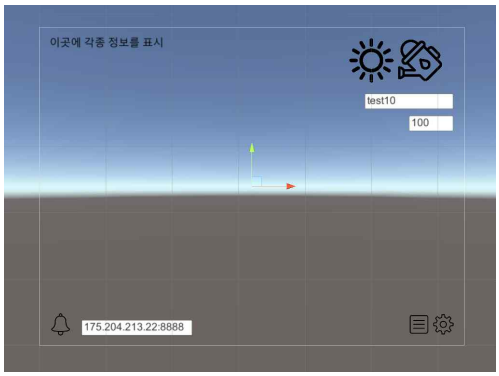
[그림5] Smart Farm 기술을 활용한 가상 농장 경영 플랫폼의 설계도

소비자는 애플리케이션을 통해 Raspberry Pi 서버로 Smart Farm에 대한 명령을 전달하고 정보를 받는다. 서버는 중간에서 클라이언트의 요청을 처리하며 그 과정에서 필요한 데이터는 저장한다. 소비자와의 소통 기능이 주를 이루는 농부의 애플리케이션은 구현하지 못했다. 소비자의 애플리케이션은 Smart Farm의 센서 값을 받아오는 것, 로그 데이터를 가져오는 것, LED를 껐다 켜는 것, LED의 상태를 확인하는 기능까지 구현했다. 추가로 네트워크의 성능을 시험하기 위해 다양한 크기의 더미 데이터를 서버로부터 전달받는 기능을 추가했다.

A. Unity



[그림6] 유니티 엔진 에디터상에서 밭의 지형을 카메라에 표시해둔 모습



[그림7] 캔버스에 UI요소들을 배치한 모습 (에디터)

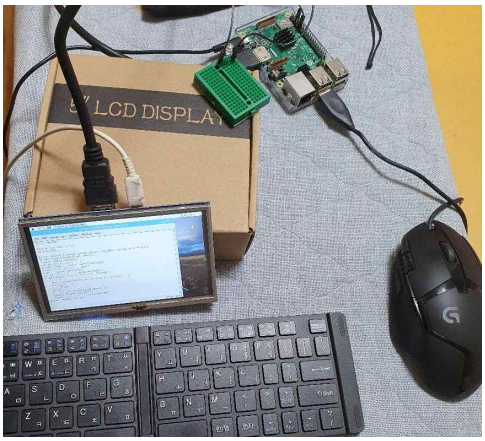
[그림8] 게임 실행 시 모습

에디터 상에서는 일단 지형과 카메라, UI 등만 배치한 후 게임이 실행될 때 Free Fab을 활용하여 작물들을 지형에 맞게 배치해주는 방식으로 애플리케이션을 구현하였다.

차후 서버에서 작물의 성장 상황에 대한 정보를 수신하여 각 작물들의 성장 상태에 맞게 각 오브젝트를 교체하여 게임적으로 표현할 수 있을 것으로 보았다.

애플리케이션 제작 도중 서버와 클라이언트 간에 HTTP 통신을 활용할 때 통신 성능이 어느 정도가 되는지 확인을 해보기로 하였고, 그에 따라 UI상의 일부 버튼의 기능을 변형하고 수정 가능한 텍스트 상자를 추가하여 매번 코드 수정을 하지 않아도 여러 종류의 HTTP 통신 테스트를 할 수 있게 구현하였다.

B. *Raspberry Pi*



[그림9] 전체적인 개발환경

Raspberry Pi에 청색 LED를 연결하고, 센서는 코드에서 임의의 값을 생성하도록 했다. HTTP 서버에 접속해 LED를 끄고 켤 수 있으며 LED의 상태, 센서 값, 로그 데이터 등을 확인할 수 있다. 기능에 대한 라우팅은 다음과 같다.

/sensor	센서 값을 불러온다.
/on	LED를 켜고 이를 데이터베이스에 기록 후 on을 표시한다.
/off	LED를 끄고 이를 데이터베이스에 기록 후 off를 표시한다.
/log	on, off 기록이 시간과 함께 나타난다.
/states	식물의 상태와 LED의 현재 상태를 알려준다.
/test10	10B의 더미 데이터
/test1000	1KB의 더미 데이터
/test100000	100KB의 더미 데이터
/test10000000	10MB의 더미 데이터
/test100000000	100MB의 더미 데이터

[표1] HTTP 서버 라우팅

로그 데이터는 모두 MySQL 데이터베이스에 시간과 함께 다음 그림과 같이 기록된다.

```
MariaDB [log]> select * from LogTable
    -> ;
+---------------------+--------+
| Date_Time           | Action |
+---------------------+--------+
| 2020-05-25 10:38:28 | on     |
| 2020-05-25 12:35:24 | off    |
+---------------------+--------+
2 rows in set (0.00 sec)

MariaDB [log]> select * from LogTable;
+---------------------+--------+
| Date_Time           | Action |
+---------------------+--------+
| 2020-05-25 10:38:28 | on     |
| 2020-05-25 12:35:24 | off    |
| 2020-05-25 12:56:31 | on     |
| 2020-05-25 12:56:35 | off    |
+---------------------+--------+
4 rows in set (0.00 sec)
```

[그림10] cmd로 MySQL에 접속한 모습

C. *실행*

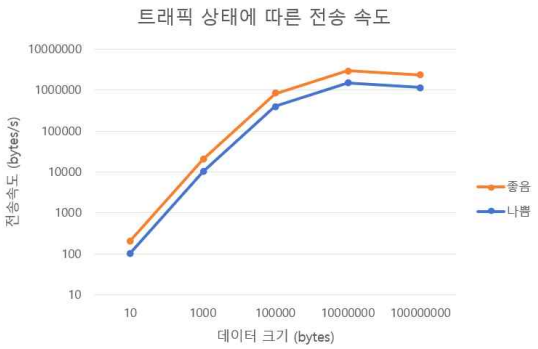
연결 방식, 데이터 크기, 트래픽 상태에 따른 전송속도를 측정했다. 데이터 크기의 경우 10B, 1KB, 100KB, 10MB, 100MB로 나눠 HTTP 전송속도의 한계를 알아보고자 했다. 같은 시간대의 LAN(천안-서울)과 LTE(천안-수원)에 대해서 측정했다. 트래픽 상태의 경우 사람이 많은 장소에서 데이터 이용이 활발한 시간대에 측정한 것과 사람이 적은 장소에서 데이터 이용이 활발하지 않은 시간대에 측정한 것을 비교한다. 지역은 천안-수원이다. 마지막으로 연결 방식의 경우 LAN(천안-서울), LTE(천안-수원), 5G(천안-수원), 내부 연결망으로 나뉜다. 모두 비슷한 시간대에 측정했다.

2. 결과 및 분석

HTTP 프로토콜을 이용하여 통신은 데이터가 전송될 때 그 크기에 상관없이 일정 크기의 헤더가 같이 전송되어야 하며, 클라이언트 측에서 요청한 후 서버에서 응답하는 방식이다. 따라서 네트워크 전파에 따른 딜레이가 각 전송 시마다 추가된다.

실험의 결과를 보면 위에 언급된 HTTP통신의 특성들로 인하여 적은 데이터를 전송하는 경우에 평균적인 데이터 전송률이 극적으로 낮아지게 되는 것을 확인할 수 있다.

또한 사람이 많은 장소에서 LTE망을 이용하는 상황과 같이 네트워크 전파에 따른 딜레이가 커지는 경우에 적은 데이터를 전송하는 과정에서 더욱 속도 저하가 심할 것으로 예상했다. 그러나 트래픽 상황에 따른 속도 저하는 데이터의 크기가 작을 때나 클 때나 비슷한 수준으로 발생한다는 것을 확인할 수 있었다.



마지막으로 내부/외부 LAN망이나 5G/LTE등의 다양한 환경에서의 전송률 저하 정도가 얼마나 차이 나는지 확인해보았다. 전송 방식에 따라 전송률 자체가 차이나는 부분은 있었으나 작은 데이터를 보낼 때 전송률 저하 정도는 비슷한 수준이었다.

Ⅲ. 결론

실험을 통해 살펴본 바와 같이, HTTP통신은 작은 데이터를 자주 주고받아야 할 때 그 효율이 크게 떨어지는 모습을 볼 수 있었다. 하지만 HTTP 프로토콜은 구조적 신뢰성이 매우 높고 이미 구현된 코드가 많아 접근성도 매우 좋다. 따라서 잦은 갱신이 필요하지 않고 데이터를 자주 주고받지 않아도 되는 애플리케이션에서는 주 데이터 교환 방식으로 HTTP 통신을 선택하는 것이 개발 비용이나 시간을 줄이는 데 큰 도움이 될 수 있을 것으로 보인다.

ACKNOWLEDGMENT

A. Unity

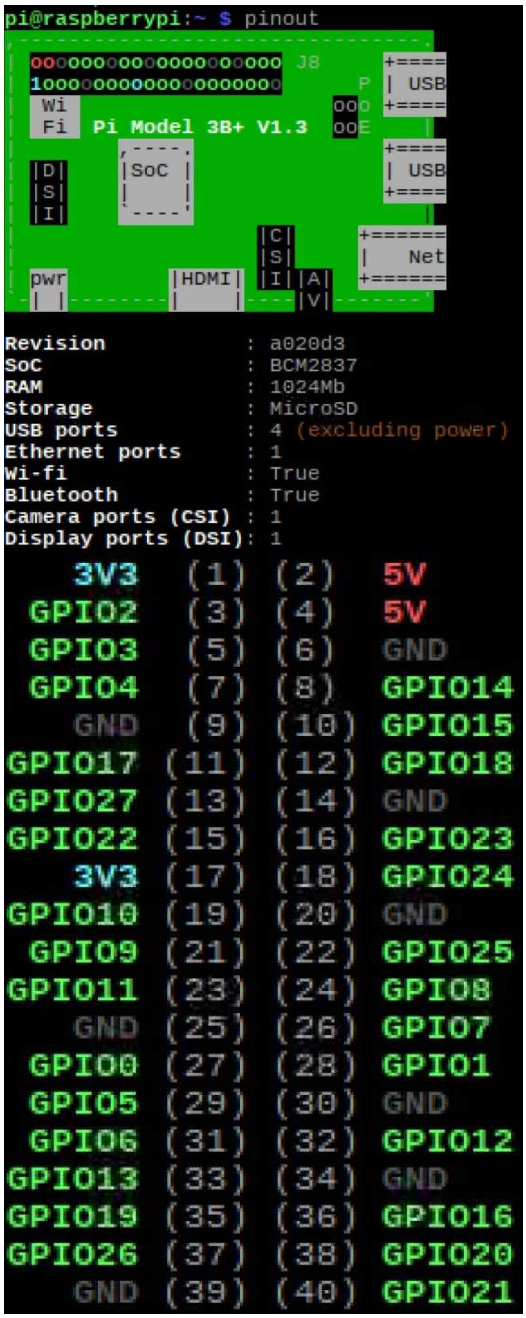
Unity는 게임 엔진이자 게임 개발 도구로써 2D/3D/VR에 이르기까지 여러 그래픽 환경과 윈도우/안드로이드/iOS/리눅스 등 다양한 운영체제에서 작동하는 게임을 간단한 사물 배치와 스크립팅으로 구현할 수 있게 해준다.

Smart Farm 애플리케이션의 기본 토대는 Unity 엔진의 에디터를 이용하여 제작하였으며, Unity 엔진에서 스크립팅 언어로 사용하는 C#을 통해 각종 사물/UI 상호작용과 네트워킹을 구현할 수 있었다.

또한 애플리케이션 빌드 시 다양한 플랫폼에서 실행할 수 있는 형태로 출력할 수 있어서 여러 네트워크 환경에서 테스트하는 데 도움이 되었다.

B. Raspberry Pi

서버 컴퓨터로 Raspberry Pi를 선택했다. Raspberry Pi는 영국 잉글랜드의 Raspberry Pi 재단이 학교와 개발도상국에서 기초 컴퓨터 과학의 교육을 증진시키기 위해 개발한 신용카드 크기의 싱글 보드 컴퓨터이다.



[그림1] pinout을 통해 확인할 수 있는 Raspberry Pi의 정보

위의 그림처럼 cmd에 pinout을 입력해 Raspberry Pi의 다양한 정보를 확인할 수 있다. 여기서 쓴 Raspberry pi의 모델은 3B+이며, 디스플레이는 HDMI 5인치 LCD 터치스크린 및 Port Forwarding을 통한 VNC Viewer를 활용했다. 개발 언어로는 Python을 선택했다.

i. Flask

HTTP 서버는 Flask 모듈로 구현했다. Python을 기반으로 하는 Flask는 특별한 도구나 라이브러리가 필요 없으므로 마이크로 웹 프레임워크라 부른다. 간결함을 추구하고 높은 자유도를 가지고 있으므로 Flask를 선택했다. Flask의 기본적인 뼈대는 다음 그림과 같다. host에 대상의 ip 주소를 입력하고 port를 입력하면 된다. (2)

```python
from flask import Flask
app=Flask(__name__)

@app.route("/")
def hi():
    return "hi"

if __name__ == "__main__":
    app.run(host='127.0.0.1', port=5555, debug=True)
```

[그림2] Flask 서버의 기본 틀

ii. PyMySQL

LED를 끄고 켜는 등 변화에 대한 로그 데이터를 저장해두는 데이터베이스가 필요했다. Python에서 MySQL 데이터베이스를 사용하기 위해 PyMySQL 모듈을 사용했다. (3)

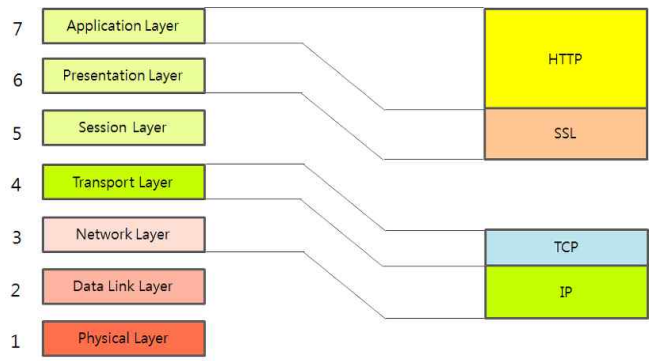
iii. GPIO

Python에서의 기본 GPIO 설정 뼈대는 다음 그림과 같다. LED가 연결된 pin 번호를 mode에 맞게 작성한 후 실행하면 LED가 켜진다. pin 번호가 BCM인지 wPi를 결정해주는 setmode 함수와 해당 pin이 입력인지 출력인지를 결정하는 setup 함수가 필수적이다. (4)

```python
import RPi.GPIO as gpio

pin=21

gpio.setmode(gpio.BCM)
gpio.setup(pin, gpio.OUT)

gpio.output(pin, True)
```

[그림3] GPIO의 기본 틀

C. *HTTP*



[그림4] HTTP 프로토콜의 OSI Layer (5)

HTTP(Hyper Text Transfer Protocol)는 WWW 상에서 정보를 주고받을 수 있는 프로토콜이다. 주로 HTML 문서를 주고받는 데에 쓰인다. TCP와 UDP를 사용하며, 80번 포트를 사용한다. HTTP는 클라이언트와 서버 사이에 이루어지는 요청/응답 프로토콜이다. 예를 들면, 클라이언트인 웹 브라우저가 HTTP를 통하여 서버로부터 웹페이지나 그림 정보를 요청하면, 서버는 이 요청에 응답하여 필요한 정보를 해당 사용자에게 전달하게 된다. HTTP를 통해 전달되는 자료는 'http:'로 시작하는 URL로 조회할 수 있다.

D. *Port Forwarding*

Port Forwarding은 컴퓨터 네트워크에서 패킷이 라우터나 방화벽과 같은 네트워크 게이트웨이를 가로지르는 동안 하나의 IP 주소와 포트 번호 결합의 통신 요청을 다른 곳으로 넘겨주는 네트워크 주소 변환의 응용이다. (6)

참 고 문 헌

(1)https://www.thekpm.com/view.php?ud=20180517162415002109 0_17&mobile=1
(2)https://velog.io/@decody/%ED%8C%8C%EC%9D%B4%EC%8D%AC-Flask%EB%A1%9C-%EA%B0%84%EB%8B%A8-%EC%9B%B9%EC%84%9C%EB%B2%84-%EA%B5%AC%EB%8F%99%ED%95%98%EA%B8%B0
(3)https://yurimkoo.github.io/python/2019/09/14/connect-db-with-python.html
(4)http://lhdangerous.godohosting.com/wiki/index.php/Raspberry_pi_%EC%97%90%EC%84%9C_python%EC%9C%BC%EB%A1%9C_GPIO_%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0
(5)https://arclab.tistory.com/120
(6)https://m.blog.naver.com/seoulworkshop/221265052717