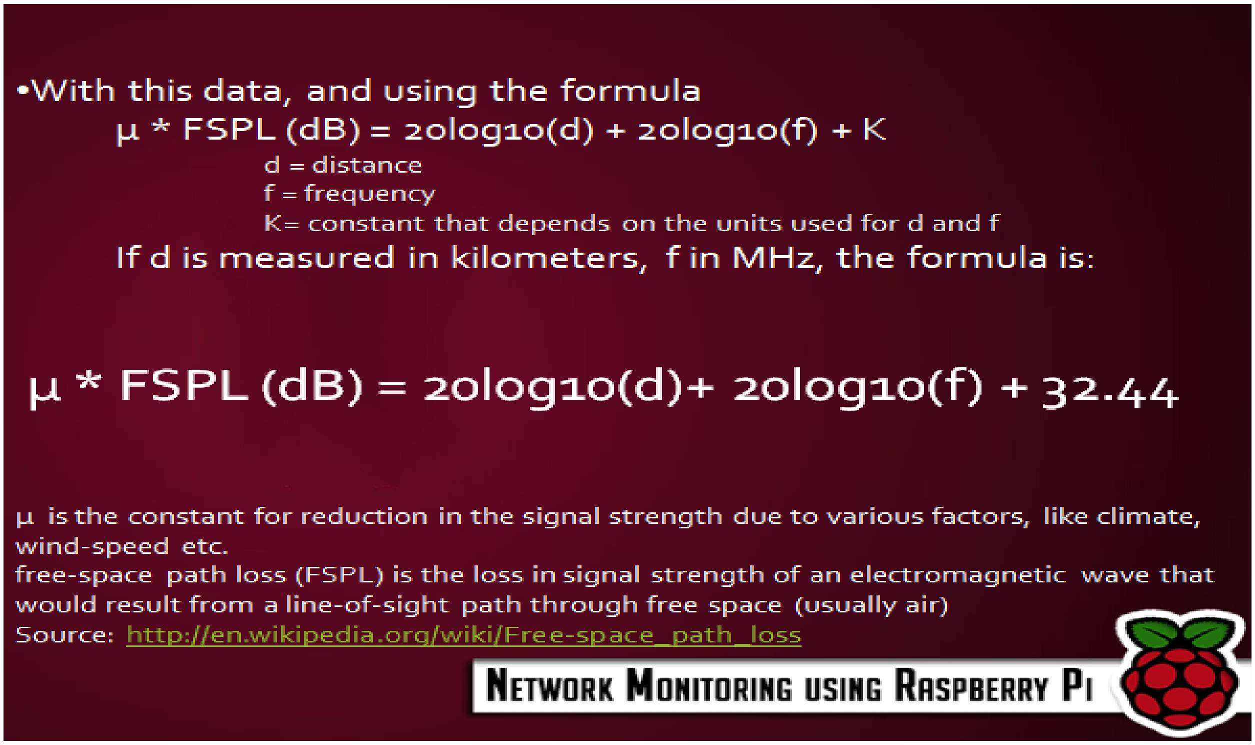
Network Monitoring Using RaspberryPi-Plotting Heat-Map

System Description

The main aim of our project is that we would be able to see the heat-map of an area, say hostel covered by RaspberryPi network. RaspberryPi are setup in different locations and they are continuously keeping track of which of the Aps are broadcasting their beacon frame. RaspberryPi writes this data on to a file and send this file on to the server where it's get parsed and stored in the database. On the server side we have a web interface which lets user interact with the database and can ask it generate the heat-map of the desired area. We are using our algorithm along with google apis to plot the heat map. We can also see the heat-map of selective devices.



•With this data, and using the formula

$$\mu * FSPL\ (dB) = 20\log10(d) + 20\log10(f) + K$$

d = distance
f = frequency
K= constant that depends on the units used for d and f

If d is measured in kilometers, f in MHz, the formula is:

$$\mu * FSPL\ (dB) = 20\log10(d) + 20\log10(f) + 32.44$$

μ is the constant for reduction in the signal strength due to various factors, like climate, wind-speed etc.

free-space path loss (FSPL) is the loss in signal strength of an electromagnetic wave that would result from a line-of-sight path through free space (usually air)

Source: http://en.wikipedia.org/wiki/Free-space_path_loss

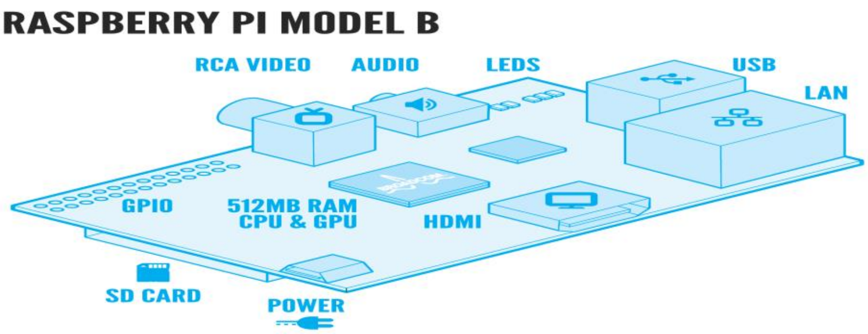
NETWORK MONITORING USING RASPBERRY PI

Only admin has the right to add or remove any device from the database via web interface. He can add users and can even delete users. Via Web Interface admin can check which of the access points are not responding and which of the RaspberryPi are not working correctly.

Expected Environment

Requirements for this project setup are:

- RaspberryPi Model B running Raspbian Wheezy Operating system



- Ubuntu Server or any Linux based server

System Requirements

Following are the requirements for the Ubuntu Server:

Hardware Requirements

- 2GB of RAM
- 10GB of Space
- No. of CPUs: Minimum 1
- Architecture : x86_64

Software Requirements

- MySQL Server
- MySQL connector module for python
  (Use this link: “http://dev.mysql.com/downloads/connector/python/”)
- SSH, SCREEN, SCP packages
- Python 2.7
- Crontab : Use sudo apt-get install cron

(*We will use cron to automate the job of backing up the database every one month*)

Following are the requirements for the RaspberryPi:

Hardware Requirements

- Wi-Fi dongle and compatible driver for Raspbian OS or linux
- 8GB Memory Card

Software Requirements

- SSH ,SSHPASS, SCREEN and SCP packages
- Python 2.7

For installing ssh, sshpass and scp use this command:

“sudo apt-get install ssh sshpass scp screen”

Installation

Setting up RaspberryPi

- Write Raspbian OS on to the memory card
- Remove the memory card(MC) and insert it into PC
- Edit cmdline.txt and add at the end of first line “ip=169.254.93.2”
- This is the Ethernet ip of the Pi and can be used to do ssh from PC.
- Use above command to install ssh, sshpass and scp, if not already install.

To install driver for GW-450D wifi dongle follow these steps(if you are using it):

1. Environment.
   ```
   # Apt-get update
   # Apt-get dist-upgrade
   # Apt-get install gcc make bc screen ncurses-dev
   ```
2. Getting the source
   ```
   # Cd /usr/src
   # Git init
   # Git clone --depth 1 http://github.com/raspberrypi/linux.git
   # Git clone --depth 1 http://github.com/raspberrypi/firmware.git
   # Cd linux
   # Gzip –dc /proc/config.gz>. Config
   # Cp -p /usr/src/firmware/extra/Module.symvers.
   ```

3. Compiling the kernel
   ```
   # Make oldconfig
   # Make clean
   # Make
   ```
4. Kernel Started
   ```
   # Make modules_install
   # Mkdir boot
   # Make INSTALL_PATH=/usr/src/linux/boot install
   # Cp /usr/src/linux/boot/vmlinuz-3.10.37+ /boot
   # Vi /boot/config.txt
   ```
   kernel = vmlinuz-3.10.37+ <- Add
5. Restart
6. Compile preparation of module
   ```
   # Ln-s /usr/src/linux//lib/modules/3.10.37+ /build
   # Cd /lib/modules/3.10.37+/build
   ```
7. confirmation of vermagic
   ```
   # Cat include/generated/utsrelease.h
   ```
   ※ Make sure you are having the output of uname-r
8. Where to get driver, http://www.planex.co.jp/support/download/gw-450d/driver_linux.shtml
   Download to the /usr/src/
9. Deployment of driver
   ```
   # Cd /usr/src
   # Unzip gw-450d_driver_linux_v3002.zip
   # Cd gw-450d_driver_linux_v3002
   # Tar jxvf mt7610u_wifi_sta_v3002_dpo_20130916.tar.bz2
   # Cd mt7610u_wifi_sta_v3002_dpo_20130916
   ```
10. Set the vendor ID
    ```
    # Cd common
    # Vi rtusb_dev_id.c
    ```
    ```
    /* module table */
    USB_DEVICE_ID rtusb_dev_id[] = {
    #ifdef MT76x0
    {USB_DEVICE(0x2019,0xab31)}, /* GW-450D */  <- add line
    {USB_DEVICE(0x148F,0x7610)}, /* MT7610U */
    {USB_DEVICE(0x0E8D,0x7610)}, /* MT7610U */
    {USB_DEVICE_AND_INTERFACE_INFO(0x0E8D, 0x7630, 0xff,
    0x2, 0xff)}, /* MT7630U */
    {USB_DEVICE_AND_INTERFACE_INFO(0x0E8D, 0x7650, 0xff,
    0x2, 0xff)}, /* MT7650U */
    #endif
    { }/* Terminating entry */
    };
    ```
11. If you use a network manager
    ```
    # Cd ..  /os/linux
    # Vi config.mk
    ```
    Change to y n - <HAS_WPA_SUPPLICANT = y
    Chang e to y n - <HAS_NATIVE_WPA_SUPPLICANT_SUPPORT = y
12. Compile
    ```
    # Cd .. /..
    # Make
    ```

13. Driver confirmation

    ```
    # Cd os/linux/
    # Modinfo mt7650u_sta.ko
    ```

    ※ It is a result of uname-r is especially vermagic

    ```
    filename: /usr/src/gw-450d_driver_linux_v3002/mt7610u_wifi_sta_v3002_dpo_20130916/os/linux/mt7650u_sta.ko
    version: 3.0.0.2
    description: RT2870 Wireless Lan Linux Driver
    author: Paul Lin <paul_lin@ralinktech.com>
    license: GPL
    srcversion: 2F72C14F7708AE4F0A65981
    alias: usb: v0E8Dp7650d * dc * dsc * dp * icFFisc02ipFFin *
    alias: usb: v0E8Dp7630d * dc * dsc * dp * icFFisc02ipFFin *
    alias: usb: v0E8Dp7610d * dc * dsc * dp * ic * isc * ip * in *
    alias: usb: v148Fp7610d * dc * dsc * dp * ic * isc * ip * in *
    alias: usb: v2019pAB31d * dc * dsc * dp * ic * isc * ip * in *
    depends:
    vermagic: 3.10.37+ preempt mod_unload modversions ARMv6
    parm: mac: rt28xx: wireless mac addr (charp)
    ```

14. Driver Deployment

    ```
    # Cp -p mt7650u_sta.ko/lib/modules/3.10.37+/kernel/drivers/net/wireless
    # Depmod-a
    ```

15. Configuration file placement

    ```
    # Cd .. / ..
    # Mkdir -p /etc/Wireless/RT2870STA
    # Cp RT2870STA.dat /etc/Wireless/RT2870STA/RT2870STA.dat
    ```

16. Describes the interface

    ```
    # Vi /etc/network/interfaces
    ```

    ※ I add the following

    ```
    allow-hotplug ra0
    auto ra0
    iface ra0 inet dhcp
    SSID to connect - wpa-ssid "your-ssid" <
    wpa-psk "your-passwd" <- password
    ```

17. Restart

    ```
    # Reboot
    ```

    NOTE: Compilation of kernel may take whole day

Setting up Ubuntu Server

It is a simple flow-navigation based web application, with several elements. The front-end has been developed using PHP, Javascript, CSS and HTML5. Back-end is on MySQL server based database server.

- **Setting back-end server**

This will install some basic packages, including php 5.4.

```
$ sudo apt-get update

$ sudo apt-get install -y vim   # Everyone likes vim, right?

$ sudo apt-get install -y build-essential

$ sudo apt-get install -y python-software-properties
```

```
# Run these 2 steps if you want php 5.4, rather than 5.3

$ sudo add-apt-repository ppa:ondrej/php5

$ sudo apt-get update
```

```
# Install the LAMP components

$ sudo apt-get install -y php5

$ sudo apt-get install -y apache2

$ sudo apt-get install -y libapache2-mod-php5

$ sudo apt-get install -y mysql-server
$ sudo apt-get install -y php5-mysql

$ sudo apt-get install -y php5-curl

$ sudo apt-get install -y php5-gd

$ sudo apt-get install -y php5-mcrypt


# Set your server name (Avoid error message on reload/restart of Apache)

$ echo 'ServerName localhost' | sudo tee /etc/apache2/httpd.conf


# Enable mod-rewrite

$ sudo a2enmod rewrite
```

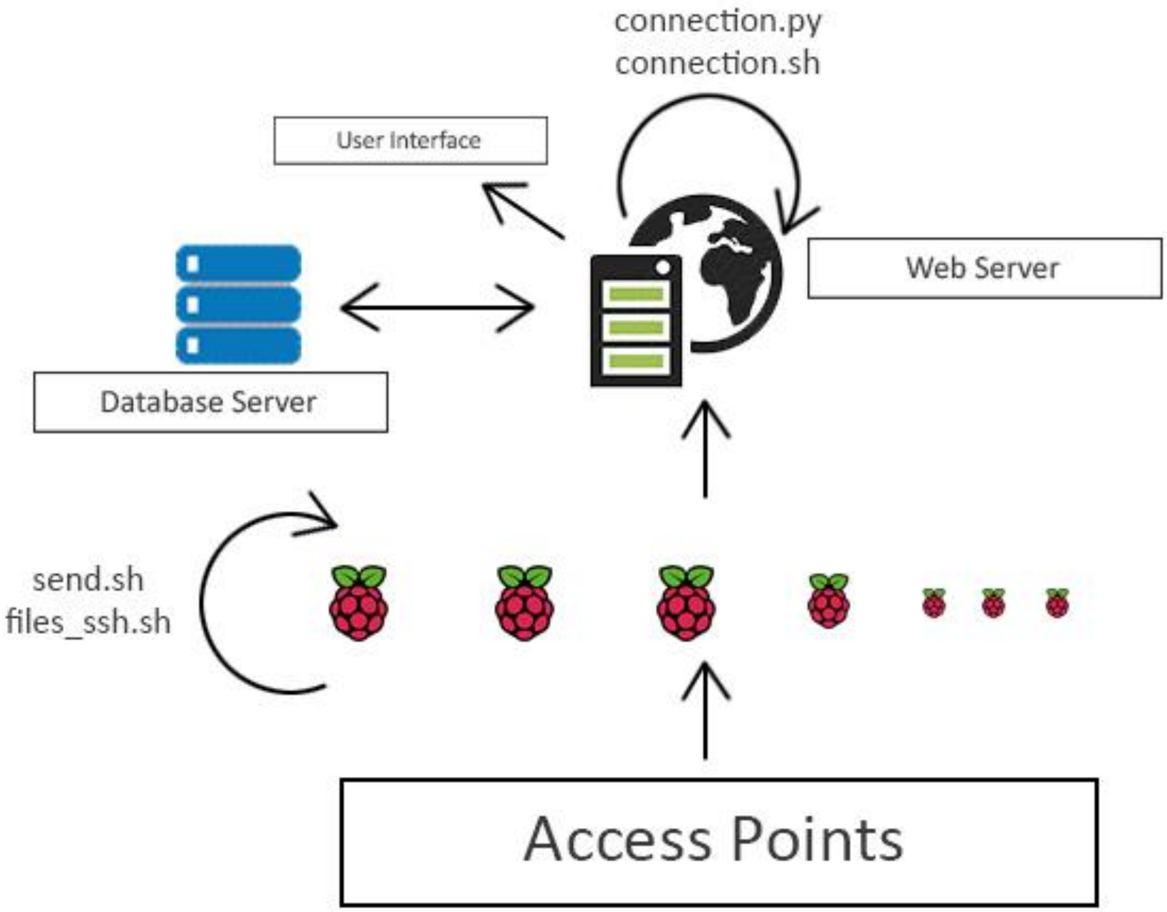
- Using MySQL and doing initial configurations

```
#login into MySQL

$ mysql –user=<username> --password=<password>


        mysql> Create database ne;

        mysql> exit;

#restore database

$ mysql -u <user> -p  <database> < <path-to-sql-file>
```

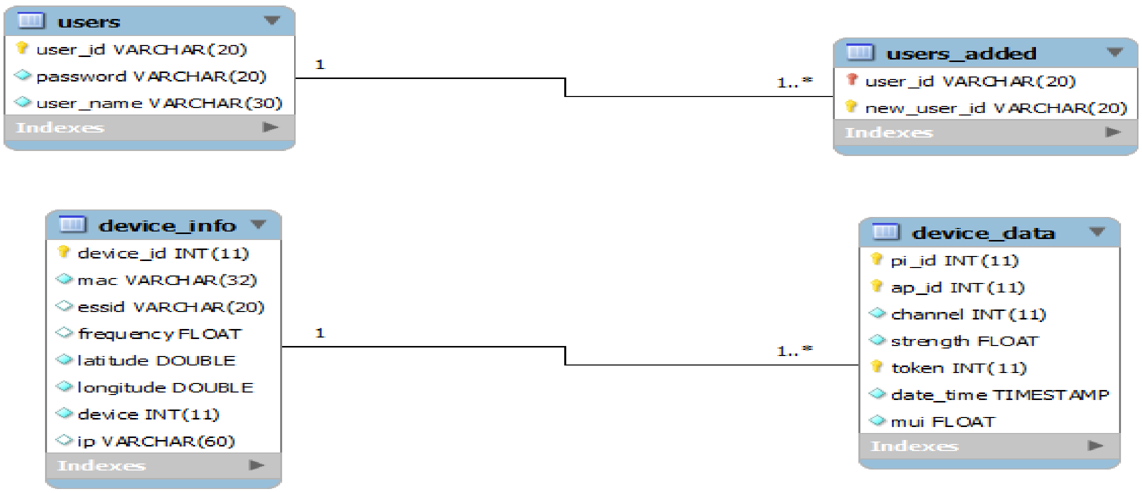
- Website Setup

```
#copy the files to /var/www/ folder and give it 777 permissions

$ sudo cp <path-to-networks-folder> /var/www/

$ sudo chmod -r 777 /var/www/
```

Network Model



Database Schema:



users
user_id VARCHAR(20)
password VARCHAR(20)
user_name VARCHAR(30)
Indexes

users_added
user_id VARCHAR(20)
new_user_id VARCHAR(20)
Indexes

device_info
device_id INT(11)
mac VARCHAR(32)
essid VARCHAR(20)
frequency FLOAT
latitude DOUBLE
longitude DOUBLE
device INT(11)
ip VARCHAR(60)
Indexes

device_data
pi_id INT(11)
ap_id INT(11)
channel INT(11)
strength FLOAT
token INT(11)
date_time TIMESTAMP
mui FLOAT
Indexes

Description of each File

Files on Server

- Connection.sh :- Main job of this script is to loop through all the files send by RaspberryPi and for each file call 'connection.py' which will parse each file and enter the data into the database. RaspberryPi is configured to send all the files to '/home/cs5113/networks/files/' folder.
- Connection.py :- This python file will parse the files send by RaspberryPi and put it accordingly in database.
- backup.sh :- This script will take backup of database and delete everything from device_data table for management purpose. This files needs to setup as cron job on server.
- database.sql:- This .sql file contains all database tables and its schema. Pls import it into mysql

Files on RaspberryPi

- send.sh :- This script generates an output file which is unique to reach Pi every 30sec. It contains device_id of Pi, token no. of file to uniquely identify each file and its Ipv4 address and data of all the APs from which it is getting the signal in a nice tabular form. It then stores them into 'files_to_send' folder.
- iwlist.py :- Main job of this file is to parse the output of 'iwlist scan' command in a more human readable format so that it is easier to parse it on the server side. This file gets called internally in send.sh
- files_ssh.sh :- This file doesn't do anything special. Its main job is to copy whatever is there in 'files_to_send' dir over to the server in a specific folder which is '/home/cs5113/networks/files/'.

Configuration

Following procedure guides how to configure RaspberryPi and Server:

Configuring RaspberryPi:

- Copy all the files from the RaspberryPi folder on to the desktop on RaspberryPi
- SSH into Pi and type “screen and hit enter”
- Edit “send.sh” ,set “device_id” to the id given by the server, ip address of server and accordingly change the interface in command starting with “ifconfig” and “iwlist” i.e if your wifi interface name is “ra0” then write “ra0” etc. You can check wifi interface name by typing “ifconfig” command in terminal.
- Set permission for “send.sh” and “files_ssh.sh”
- Execute send.sh and files_ssh.sh via root user as background process using “&”.
- Hit Ctrl-A and then Ctrl-D to detach from screen. To connect again to screen use “screen –r”
- Now both the process are running and you can close your ssh session.
- Files generated by “send.sh” are put into “files_to_send” dir and “files_ssh.sh” sends those files to the server using scp.

Configuring Server:

- Copy all the files from Server folder on to the desktop
- Import the database file by running the command:
  “mysql –u root –p’password’ –h localhost ne < database.sql”
- Type “crontab –e” and write
  “30 8 1 * * /bin/bash /home/cs5113/networks/backup/backup.sh” at the end of file to automatically backup of database every one month and delete all entries from “device_data” table. Backup is stored in backup folder only.
- Set permission for “connection.sh”

- And run it in background via Screen command and & just like we did in configuration of RaspberryPi.
- Set up the webserver and it's all done.

TODOs For Further Improvements

- To reduce the cost of setup drastically we could use laptops to run our script and send the data to server instead of RaspberryPi. Problem with this approach is that user needs to manual have to give their location so that our heat-map plotting algorithm can work perfectly.
- Presently, this setup works for 2D model but this approach could be extended for plotting heat-map inside 3D buildings.

For further guidance and support, please contact us at:
Vikash Kumar: cs11b038@iith.ac.in
Deepak Bhardwaj: cs11b009@iith.ac.in