

# **Review of SURF-2017 --- Server part**

---

**By: Zhenghang Zhong (Klaus)**

# Indoor Localization --- Background problems

---

Key words:

- RSS: Received signal strength (usually between 0dbm and -90dbm)
- RSSI: Received signal strength Indication (adjusted RSS presented as a positive value )
- SSID (ESSID): Service Set Identifier (changeable)
- BSSID: mac address of the service (unique)
- AP: Accessing point
- TOA: Time of arrival
- TDOA: Time difference of arrival
- AOA: Angle of arrival

# Why not GPS

- No direct line of sight between satellites and receives.

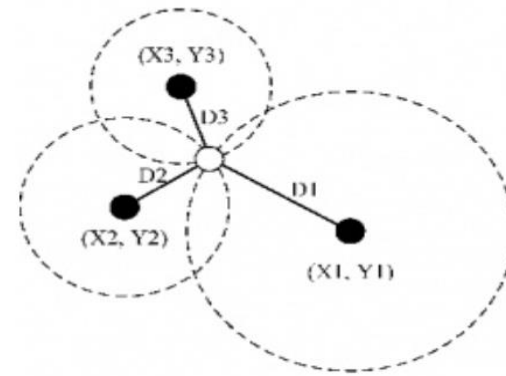
# Why choose RF fingerprinting schemes

- Widespread network devices --- easily deployable.
- Reasonable performance
- Affordable cost

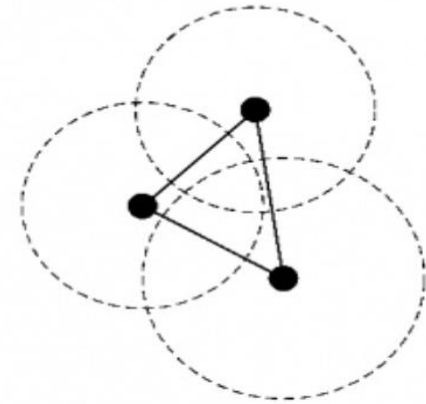
# Algorithms used in Indoor Localization

## 1. Triangulation algorithm

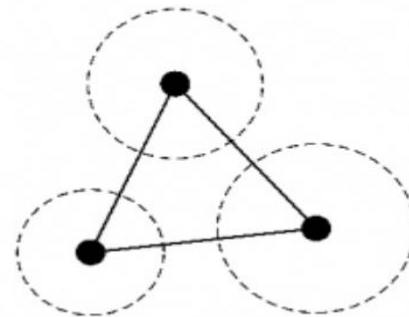
- Unknown node
- Reference node



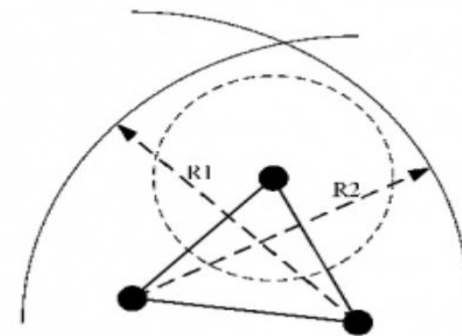
(a)



(b)



(c)



(d)

# Algorithms used in Indoor Localization

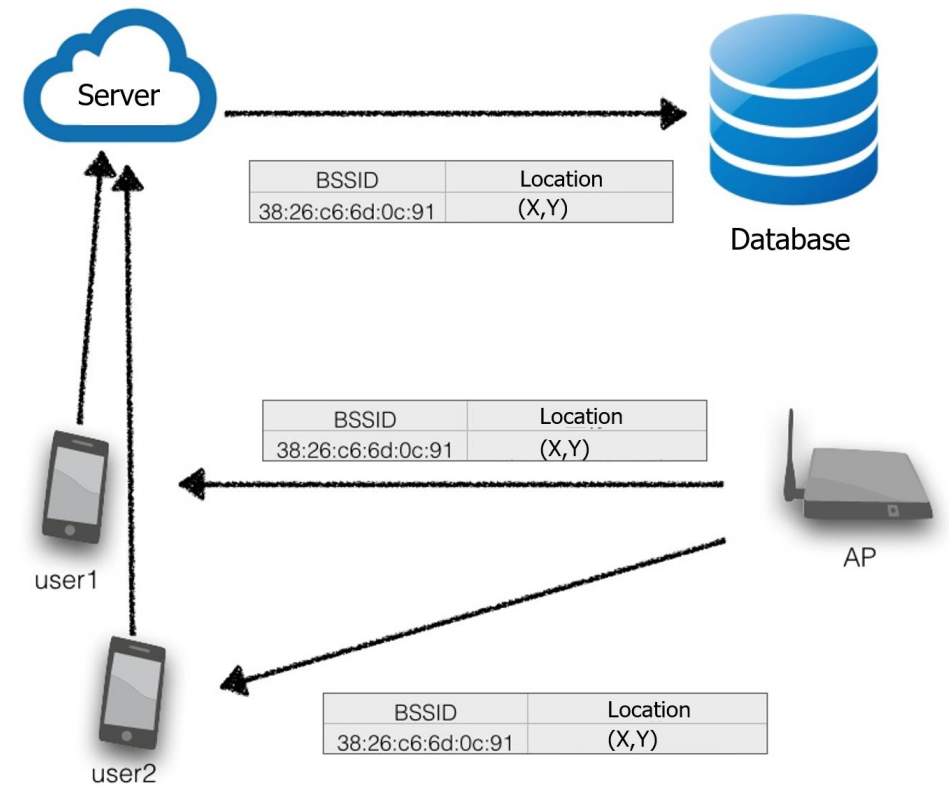
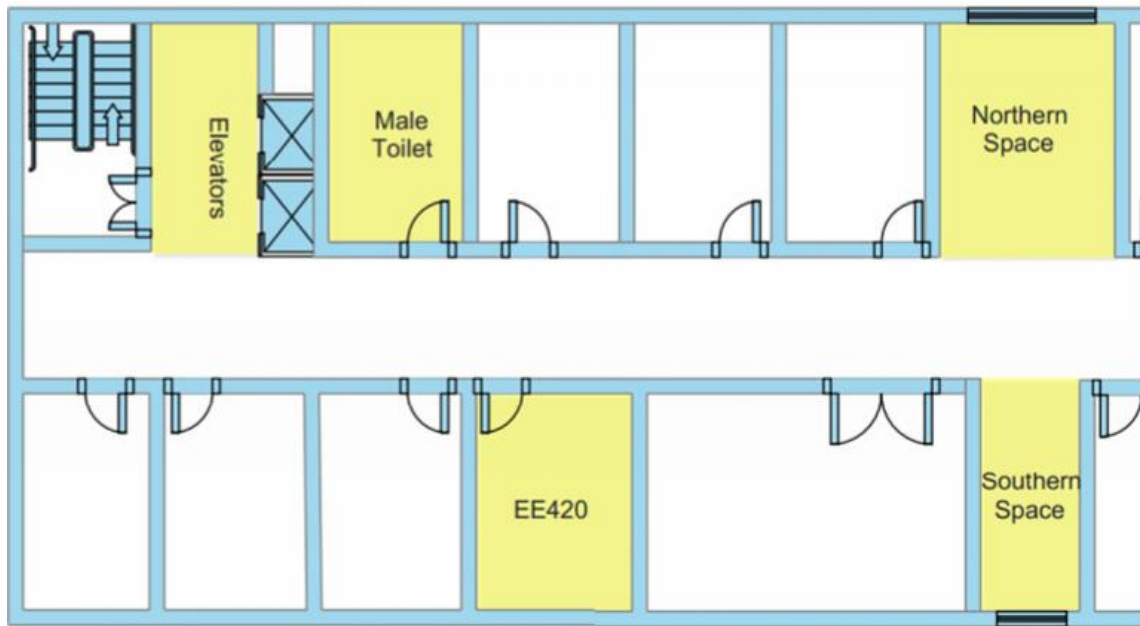
---

## 2. Fingerprint localization algorithm

1. Data collection and storage.
2. Train the neural network and model generation
3. Testing.

# Fingerprint localization algorithm

## 1. Data collection and storage.



# Database

ID	BSSID	Level	Room	Model	Time
963757	38:46:08:c9:87...	-85	1	OPPO A57	2017-08-16 10:...
963758	9c:50:ee:30:3d...	-110	1	OPPO A57	2017-08-16 10:...
963759	9c:50:ee:3f:9e...	-110	1	OPPO A57	2017-08-16 10:...
963760	b0:75:d5:80:8...	-110	1	OPPO A57	2017-08-16 10:...
963761	9c:50:ee:3f:9c...	-110	1	OPPO A57	2017-08-16 10:...
963762	9c:50:ee:3f:73...	-110	1	OPPO A57	2017-08-16 10:...
963763	9c:50:ee:30:3f...	-110	1	OPPO A57	2017-08-16 10:...
963764	9c:50:ee:3f:8a...	-110	1	OPPO A57	2017-08-16 10:...
963765	9c:50:ee:3f:99...	-110	1	OPPO A57	2017-08-16 10:...
963766	9c:50:ee:3f:99...	-110	1	OPPO A57	2017-08-16 10:...
963767	9c:50:ee:3f:71...	-84	1	OPPO A57	2017-08-16 10:...
963768	9c:50:ee:3f:8d...	-110	1	OPPO A57	2017-08-16 10:...
963769	9c:50:ee:3f:74...	-110	1	OPPO A57	2017-08-16 10:...
963770	d4:b1:10:ac:62...	-110	1	OPPO A57	2017-08-16 10:...
963771	a8:58:40:59:a...	-110	1	OPPO A57	2017-08-16 10:...
963772	4c:e6:76:64:df...	-110	1	OPPO A57	2017-08-16 10:...
963773	9c:50:ee:30:36...	-110	1	OPPO A57	2017-08-16 10:...
963774	9c:50:ee:3f:74...	-93	1	OPPO A57	2017-08-16 10:...
963775	9c:50:ee:3f:9e...	-75	1	OPPO A57	2017-08-16 10:...
963776	9c:50:ee:3f:90...	-110	1	OPPO A57	2017-08-16 10:...
963777	9c:50:ee:3f:a2...	-110	1	OPPO A57	2017-08-16 10:...
963778	9c:50:ee:3f:9e...	-110	1	OPPO A57	2017-08-16 10:...
963779	ac:4e:91:61:21...	-110	1	OPPO A57	2017-08-16 10:...
963780	9c:50:ee:3f:a0...	-110	1	OPPO A57	2017-08-16 10:...
963781	d8:c8:e9:52:da...	-110	1	OPPO A57	2017-08-16 10:...
963782	9c:50:ee:3f:90...	-110	1	OPPO A57	2017-08-16 10:...
963783	9c:50:ee:3f:a2...	-110	1	OPPO A57	2017-08-16 10:...
963784	ac:4e:91:49:fc:c1	-110	1	OPPO A57	2017-08-16 10:...
963785	9c:50:ee:3f:98...	-110	1	OPPO A57	2017-08-16 10:...
963786	9c:50:ee:3f:90...	-110	1	OPPO A57	2017-08-16 10:...
963787	9c:50:ee:3f:90...	-66	1	OPPO A57	2017-08-16 10:...
963788	9c:50:ee:3f:9d...	-110	1	OPPO A57	2017-08-16 10:...
963789	9c:50:ee:3f:9c...	-110	1	OPPO A57	2017-08-16 10:...
963790	ac:4e:91:49:fd...	-110	1	OPPO A57	2017-08-16 10:...
963791	9c:50:ee:3f:8a...	-110	1	OPPO A57	2017-08-16 10:...
963792	cc:34:29:6d:f3...	-89	1	OPPO A57	2017-08-16 10:...
963793	b0:75:d5:5f:d3...	-87	1	OPPO A57	2017-08-16 10:...
963794	a8:58:40:59:ac...	-110	1	OPPO A57	2017-08-16 10:...
963795	9c:50:ee:3f:91...	-110	1	OPPO A57	2017-08-16 10:...
963796	9c:50:ee:3f:9f:21	-110	1	OPPO A57	2017-08-16 10:...
963797	9c:50:ee:3f:93...	-110	1	OPPO A57	2017-08-16 10:...

# Fingerprint localization algorithm

## 2. Train the neural network and model generation

This part is carried out by Jeff Wong, with the collected data it is able to generate a DNN-based indoor localization model.

## 3. Testing.

- client (Android mobile phone with **WiFiScanner**, or **Raspberry Pi**)
- server with localization algorithm
- Communication through hyper text transfer protocol (**HTTP**)



# WampServer

---



- Quick start up and prototyping
- Complete environment

# Anaconda (set up the environment)

---



- Python + R, data science + machine learning
- multi-platform --- Windows, Linux, and Mac OS X
- Libraries management

# Download and Installation

 Windows

 macOS

 Linux

Anaconda 5.2 For Windows Installer

**Python 3.6 version \***

 **Download**

[64-Bit Graphical Installer \(631 MB\)](#) 

[32-Bit Graphical Installer \(506 MB\)](#)

**Python 2.7 version \***

 **Download**

[64-Bit Graphical Installer \(564 MB\)](#) 

[32-Bit Graphical Installer \(443 MB\)](#)

# Conda testing

---

Just like **git** command, the **conda** is both packages manager and environment manager. After the installation, to test if **conda** works well.

```
1 Windows + R
2 cmd
3 conda --version ::it return the version of conda
```

```
1 conda update conda
2 ::check the update detials and confirm
3 y
```

# Build up the first environment

```
1 | conda create --name <environment_name> python=3.5
```

```
Solving environment: done
```

```
## Package Plan ##
```

```
environment location: C:\A_programs\Anaconda\envs\testing
```

```
added / updated specs:
```

```
- python=3.5
```

```
The following NEW packages will be INSTALLED:
```

```
certifi:          2018.4.16-py35_0  
pip:              10.0.1-py35_0  
python:          3.5.5-h0c2934d_2  
setuptools:      39.2.0-py35_0  
vc:              14-h0510ff6_3  
vs2015_runtime: 14.0.25123-3  
wheel:           0.31.1-py35_0  
wincertstore:    0.2-py35hfebbdb8_0
```

# Packages management

---

```
1 conda list ::display all installed packages
2 conda search <package_name> ::check if packages are available
```

For some packages that cannot be installed by `conda`, we could use `pip` which has been prepared in Anaconda.

# Web frameworks

---

- **Django** is popular but heavy and complex.
- **web.py** is light but not maintained now.
- **tornado** has not much libraries as flask, may need some repetitive work.
- **flask** is light, popular, flexible, and extensible.



# Flask

web development,  
one drop at a time

- microframework for python
- Based on *Werkzeug* (The python WSGI Utility Library )  
and *Jinja2*( a full featured template engine for Python )
- Flexible, extensible.



# More reasons

- flask + DL framework
- Python 3.5
- POST and GET methods

# RESTful framework

**Representational State Transfer (REST)** is an architectural style that defines a set of constraints and properties based on [HTTP](#).

- **GET** -- Provides a read only access to a resource.
- **PUT** -- Used to create a new resource.
- **DELETE** -- Used to remove a resource.
- **POST** -- Used to update an existing resource or create a new resource.

### HTTP methods

Uniform Resource Locator (URL)	GET	PUT	PATCH	POST	DELETE
<p><b>Collection, such as</b>  <a href="https://api.example.com/resources/">https://api.example.com/resources/</a></p>	<p><b>List</b> the URIs and perhaps other details of the collection's members.</p>	<p><b>Replace</b> the entire collection with another collection.</p>	<p>Not generally used</p>	<p><b>Create</b> a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.<sup>[17]</sup></p>	<p><b>Delete</b> the entire collection.</p>
<p><b>Element, such as</b>  <a href="https://api.example.com/resources/item17">https://api.example.com/resources/item17</a></p>	<p><b>Retrieve</b> a representation of the addressed member of the collection, expressed in an appropriate Internet media type.</p>	<p><b>Replace</b> the addressed member of the collection, or if it does not exist, <b>create</b> it.</p>	<p><b>Update</b> the addressed member of the collection.</p>	<p>Not generally used. Treat the addressed member as a collection in its own right and <b>create</b> a new entry within it.<sup>[17]</sup></p>	<p><b>Delete</b> the addressed member of the collection.</p>

# Installation for Windows

```
1 cd <enviroment_name>\Scripts  ::all below packages should be installed under
   folder Scriptes
2 pip install flask
3 pip install msgpack
4 pip install flask-login
5 pip install flask-openid
6 pip install flask-sqlalchemy
7 pip install sqlalchemy-migrate
8 pip install flask-whooshalchemy
9 pip install flask-wtf
10 pip install flask-babel
11 pip install guess_language
12 pip install flipflop
13 pip install coverage
```




















# Installation for Linux, OS X or Cygwin

```
1  :::#You'd better try pip3 instead of pip, in case both python2 an python3 exist
2
3  cd <enviroment_name>\Scripts  :::all below packages should be installed under
   folder Scriptes
4  pip install falsk
5  pip install msgpack
6  pip install flask-login
7  pip install flask-openid
8  pip install flask-mail
9  pip install flask-sqlalchemy
10 pip install sqlalchemy-migrate|
11 pip install flask-whooshalchemy
12 pip install flask-wtf
13 pip install flask-babel
14 pip install guess_language
15 pip install flipflop
16 pip install coverage
```

# Copy and paste relevant files

📁 Papers	Add Paper about Hierarchical Classification
📁 SURF_DATA	Revert "Adding database part (using sqlite3) and relevant code"
📁 algorithm	Update dataset
📁 android	Wifi_Scanner
📁 data_collection	Upload poster
📁 flask	Merge pull request #8 from ZzhKlaus/master
📁 img	Add oppo test acc
📁 wifiScanner_comb_flask	Add oppo test acc
📄 .gitignore	Find problem of autoencoder
📄 Data_Description.md	Create Data_Description.md
📄 Poster.pdf	Upload poster
📄 README.md	Update README.md

# Copy and paste relevant files

 <code>__pycache__</code>	Use oppo to collect data
 <code>app</code>	Use oppo to collect data
 <code>trained_model</code>	Adding Model and Time characters in .csv and code
 <code>config.py</code>	flask&android_file
 <code>db_create.py</code>	Revert "Adding database part (using sqlite3) and relevant code"
 <code>db_downgrade.py</code>	Revert "Adding database part (using sqlite3) and relevant code"
 <code>db_migrate.py</code>	Revert "Adding database part (using sqlite3) and relevant code"
 <code>db_upgrade.py</code>	Revert "Adding database part (using sqlite3) and relevant code"
 <code>fingerprints.db</code>	Adding the DB part and relevant code
 <code>function_version.py</code>	flask&android_file
 <code>main_build_DB.py</code>	Revert "Adding database part (using sqlite3) and relevant code"
 <code>main_user.py</code>	Merge pull request #8 from ZzhKlaus/master
 <code>mapping.csv</code>	flask&android_file
 <code>mapping.py</code>	Add Timer function in android
 <code>model.py</code>	flask&android_file
 <code>oneTime.csv</code>	Adding Model & Time characters.
 <code>templist.csv</code>	Use oppo to collect data
 <code>xxx.csv</code>	Adding the DB part and relevant code
 <code>xxx_stores_all_RSS.txt</code>	flask&android_file

# Install TensorFlow and keras to run main\_user file

A quick test of the file `main_user.py` need the environment of **TensorFlow (CPU version, while GPU version is optional)** and **Keras**, which could be installed by typing below commands.

```
1 pip install tensorflow
2 pip install kearas
3 pip install pandas
4 pip install sklearn
5 pip install matplotlib
```

Tips: `conda install anaconda`



# Address setting

- Localhost address :

- IPv4 address

```
C:\Users\zheng>ipconfig

Windows IP 配置

无线局域网适配器 本地连接* 2:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

无线局域网适配器 本地连接* 3:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

无线局域网适配器 WLAN:

    连接特定的 DNS 后缀 . . . . . : DHCP HOST
    本地链接 IPv6 地址. . . . . : fe80::21a4:30b:cbc7:d577%4
    IPv4 地址 . . . . . : 192.168.1.102
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . : 192.168.1.1

以太网适配器 蓝牙网络连接:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :
```



# Flask Tutorial

```
app/  
  __init__.py  
  routes.py  
  microblog.py
```



*app/\_\_init\_\_.py*: Flask application instance

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

*app/routes.py*: Home page route

```
from app import app  
  
@app.route('/')  
@app.route('/index')  
def index():  
    return "Hello, World!"
```

```
microblog.py x  
1 from app import app  
2  
3 if __name__ == "__main__":  
4     app.run(host='127.0.0.1', debug=True)
```

# Flask Tutorial

```
>python microblog.py
```

```
* Debug mode: on  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 187-650-747  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

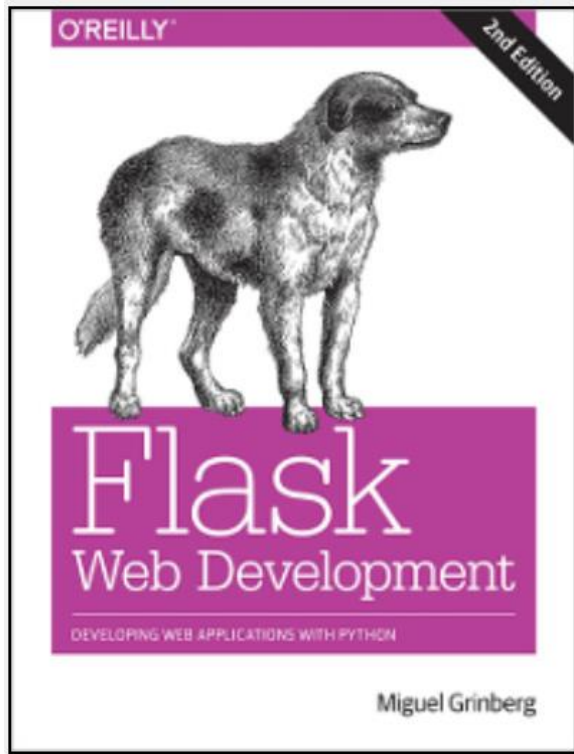


127.0.0.1:5000

XJTLU GitHub SURF SURF\_2018 Google IEEE IEEE Xplore Digital NumPy Refere

Hello, World!

# Flask Tutorial



## The Flask Mega-Tutorial Part I: Hello, World!

December 6 2017

Posted by [Miguel Grinberg](#) under [Flask](#), [Programming](#), [Python](#)

[Tweet](#) [Like](#) [G+](#) [in Share](#)



# Improvements

## 1. RESTful --- GET method

- Time Limitation
- Be short of hands
- Android, flask
- laptop, phone

## 2. flask-restful library

```
1 | pip install flask-restful
```

## 3. Robustness

- Irregular inputs
- Multi-user (multi-phones)
- Multithreading

**END**