WIFIOLOGY

To Illuminate The Current State Of Wireless Connectivity In Order To Inform And Optimize Users’ Networking Experience

Meet the Team

- Baiyu Chen - Front End/ Selenium test
- Peng Jiang - Front End
- Ryan Campbell - Android
- Robert Cope - Back end /node-js integration / 802.11 research / API / Swiss Army Knife
- Jason Nguyen - Back end / email notification integration / API
- Jasper Niemeyer - Project management, meeting minutes, milestones, initial heroku deployment, nodejs integration

Initial Idea, Motivation and Pivot

- Wanted to know if certain study areas were crowded before trying to meet there
- Use WiFi traffic to estimate the number of people in the vicinity of our listening device
- This alone would have only really counted as one or two features + login
- Add features to visualize and export data
- With some additional work our app now has the potential to serve a broader commercial market giving network admins:
    - Monitor network traffic in real time
    - Compare to historical data of that network
    - Diagnose the health of network equipment
    - Identify suspicious mac addresses on the network

Project Management & Meetings

- Met every week on Monday evenings
- Used a Kanban process (this is little-a “agile”)
  - No sprints
  - Capacity to pick up tasks as they are created
  - Retrospectives at the beginning of each meeting
  - Task run down at the end of each meeting
  - Continuous Integration, Continuous Deployment - heroku, travis CI
- Tasks managed using Trello
  - Great in the initial research phase
  - Became cumbersome towards the end of the project as tasks kept changing in scope
    - This might be less of a problem if we were more experienced and could better anticipate our requirements
- Communication using Groupme

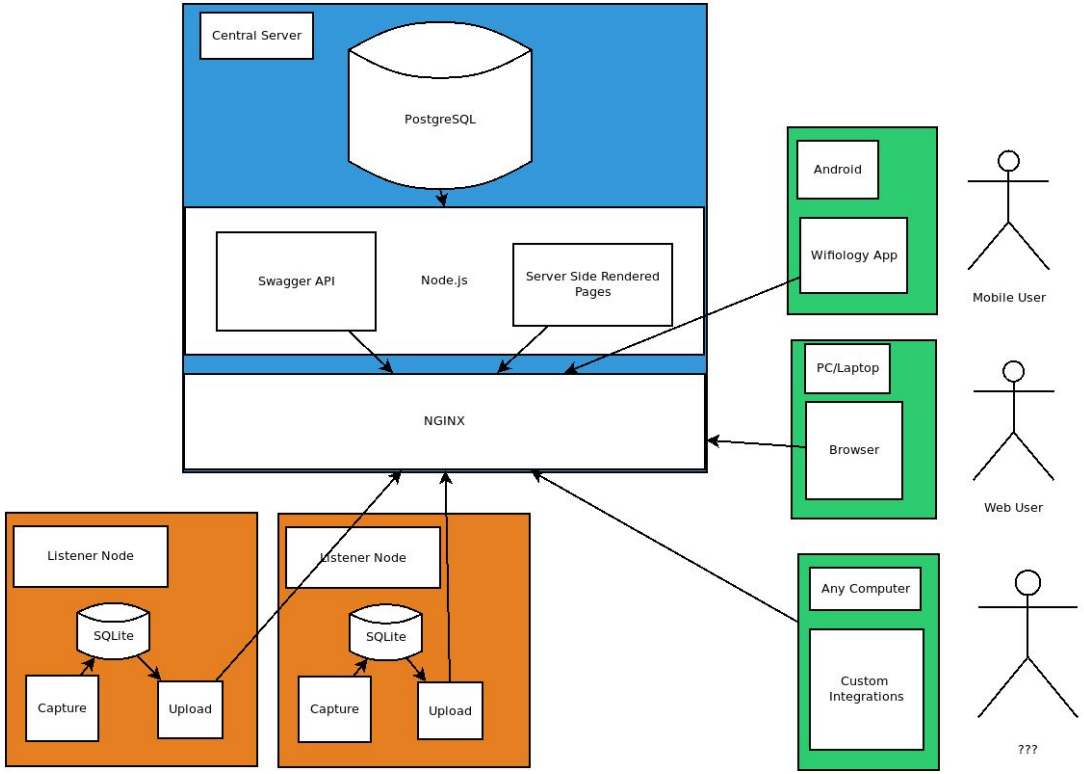
Challenges

- Teamwork challenges:
  - Diverse range of experience among team members
- Tech challenges
  - 802.11 - Learning how to decipher 802.11 to give us what info we needed
  - Heroku DB limitations - Hobby mode had it's limits
- Modeling: What does busy mean?
  - With all the given data, how do we analyze it?
- Node.js: It is hard for everybody to get start with
  - Not everyone is familiar with how to wield the async secret sauce.

High Level Architecture

- Users have many physical 802.11 listening devices with supported 802.11 card
  - Python listener node codebase sniffs 802.11 traffic, analyzes, caches in SQLite DB
  - Python listener code batch uploads from SQLite DB to central server
    - Upload via REST API (Swagger)
- Node.JS central server in front of PostgreSQL database
  - All listener nodes, corresponding API keys, etc. tracked and registered here
  - Provides API for listener nodes to upload to, API for other consumers to utilize
    - OpenAPI 2.0/Swagger
  - Provides web frontend for monitoring, analysis
- Android application (Java)
  - Utilizes central server REST API to provide native mobile data tracking and analysis

High Level Architecture (Cont.)



Tools (Not Exhaustive)

Tools, Languages & Resources Continued

Central Server

- NodeJS
- ExpressJS
- OpenAPI 2/Swagger
- jQuery
- Bootstrap 4
- PostgreSQL 10+
- Plpgsql & SQL
- Mocha & Chai
- Selenium

Listener Node

- Python
- Pcapy, Scapy, dpkt
- requests
- SQLite3
- Unittest
- Bottle
- Eventlet

Android App

- Android Studio
- Swagger Client
- Java for Classes
- XML for Layouts
- Material.io
- Volley

Operations

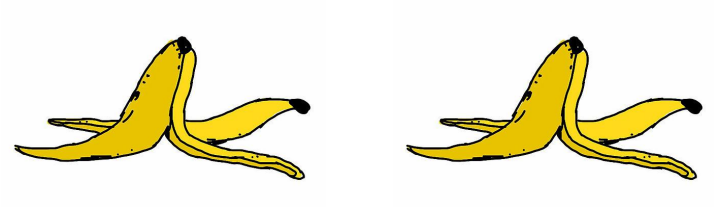
- RamNode
- Nginx
- Pm2
- Munin (Monitoring)

Trello

- Coordination is challenging
- Hard to keep up with scope
- Hard to keep up with changes in requirements
- Cards become deprecated as feature requirements changed
- Process is new to most team members

Groupme

- Overall rating: 3 stars for being useful + 1 star for not being slack
  - It's a step up from SMS group messaging
  - Can't create group DMs or unhide message threads
  - Webhhooks, Yay!
    - Used IFTTT to ping group anytime theres a pull request on github

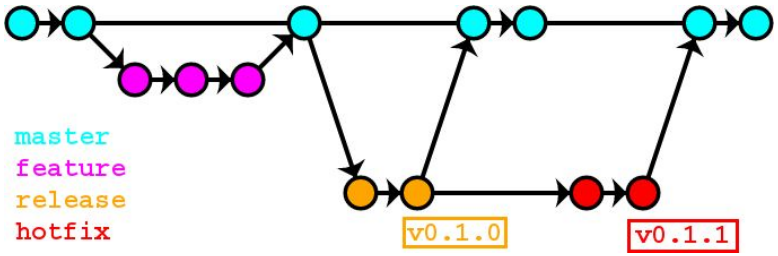


- Overall rating: 2 Banana Peels out of a possible 5 stars
- JS itself has gone through many evolutions
  - Easy to find bad examples online
  - Without prior knowledge of current best practices hard to know correct answer
- Unfamiliar programming paradigm
  - Promises, callbacks and Async awaits are foreign to most students at this level
- More difficult to learn than it first appeared

Ramnode

- Inexpensive virtual machines, good performance
- Just a VM
- Need to know how to provision a VM (securely) for a node app

- Overall rating, 3 out of 5 stars plus an avacado fella
- Slight learning curve but relatively easy to manage
  - Gitflow workflow mode
  - Compartmentalization of code
    - team members working on different files to avoid merge conflicts
  - We used tags like pros :)



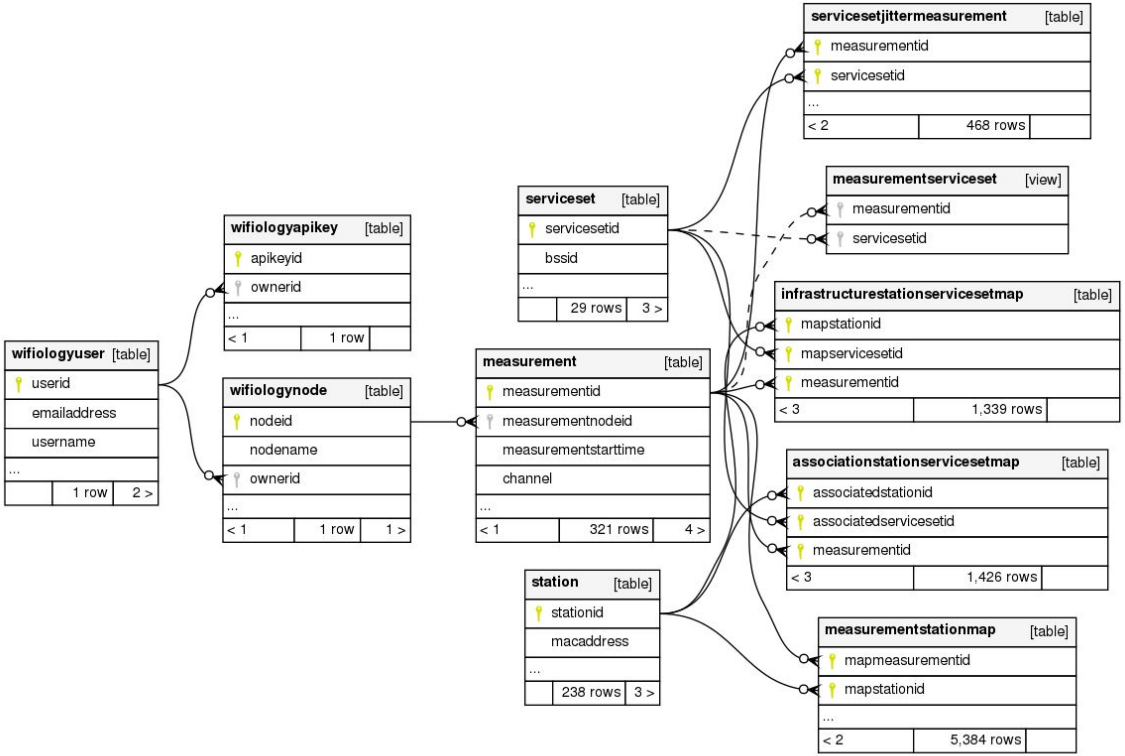
heroku ⭐⭐

- Overall rating: 2 stars
- Easy to get set up and running
- Nice CI tools, automatic build process and github integration
- Free tier is way too limited for the size of our DB
- Billing is deliberately opaque -> paid tier not ideal either
  - Could cost a company a lot of money if a dev's were spinning up side projects on heroku at company expense. (an actual problem)
  - Could cost a hobby developer a lot if their app suddenly scales -> upside is that Heroku could in theory handle that scaling
- UX is pure Bait & Switch
- Monitoring and profiling

- Overall rating 4 out 5 ‘**SELECT *** s
- Excellent combination of traditional SQL + document DB features (like JSONB)
- Migrations are easy (transactional DDL)
- Easy integration with Heroku
- Using plpqsql is only ok - hard to debug but makes queries look nice

- SQLite was also used on client: Best swiss army knife ever

PostgreSQL Schema (NodeJS, Compact)

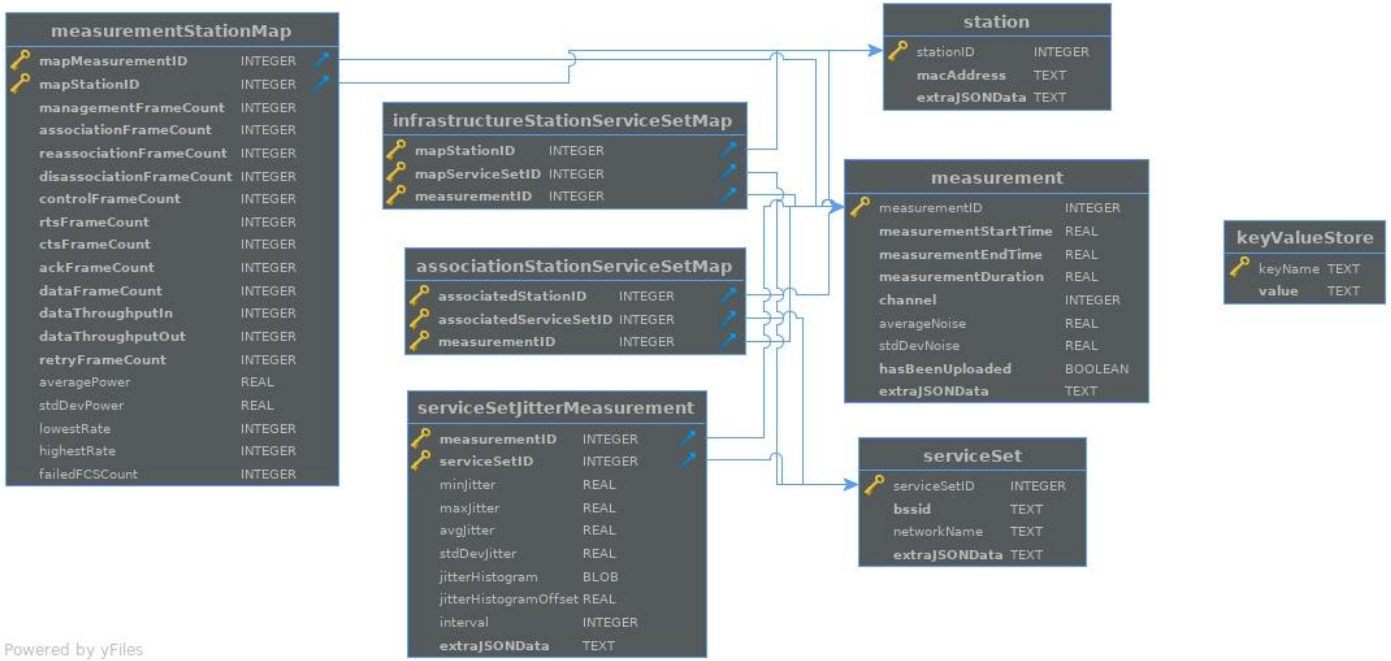


featureflag	[table]
featureflagkey	varchar[512]
featureflagvalue	jsonb[2147483647]
< 0	0 rows	0 >

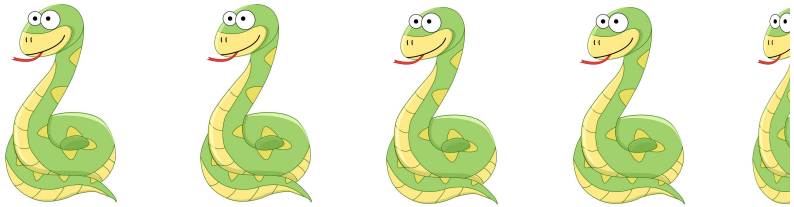
migrations	[table]
id	serial[10]
name	varchar[255]
run_on	timestamp[29,6]
< 0	5 rows	0 >

Generated by SchemaSpy

SQLite Schema



python™



- 4.5 Sneks out of 5 Sneks
- Easy for the team to pick up, good tooling and libraries
- Slow at times when running on the Raspberry Pi
- *(Editorialization: Maybe Python instead of NodeJS for the class next time?)*

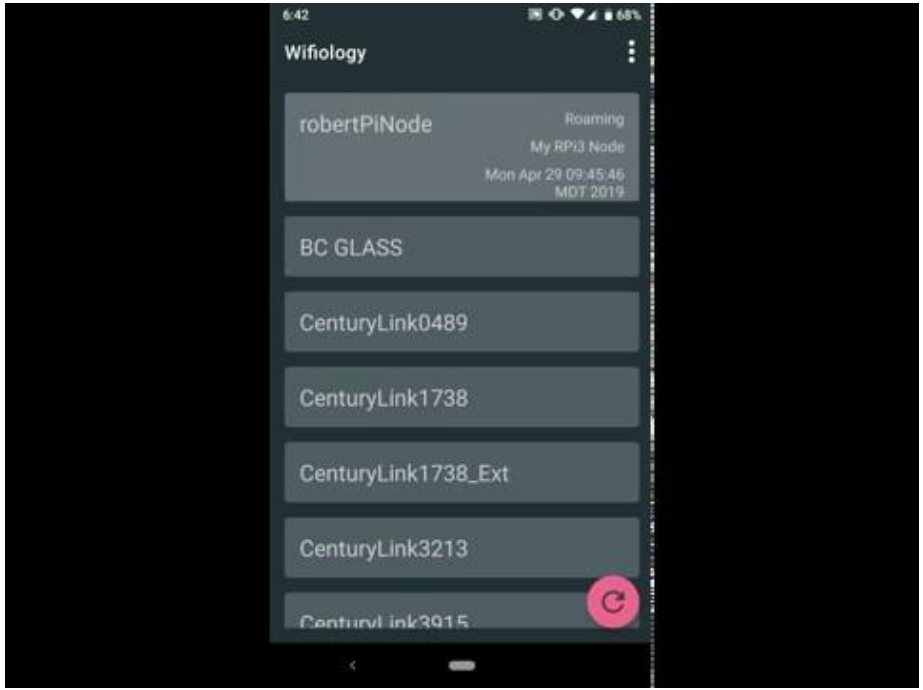
Swagger

- Open Source Software Framework to help develop a RESTful API
- Really easy and quick to produce an API to test and consume
- Likely the future of almost all new REST APIs (makes REST less terrible)

- Used on Linux in Chrome OS
- Required terminal wizardry to get working (gradle -_-)
- Randomly crashing for no reason
- Otherwise solid, based on intellij & has the darcula theme
- Ability to deploy on device was AMAZING
  - Normally testing done in emulator or over adb connection
  - Deployed from linux vm to android container in Chrome OS
  - Was instant, smooth, and consistent

- App itself: Material Themeing, Volley HTTP Library, RecyclerViews

Android Demo



Frontend

- Deciding the theme. It has to be cool, easy to read and user-friendly
- All in one page including login and register at first
- The web page contains too many junk javascript and css files and it is hard to cooperate with backend
- Get rid of the all-in-one page and start over again with a concise looking

Give me a

Login In

Sign In

Wifiology

Error 404: Group Name Not Found

An Unexist Group From CU Boulder

Login In

Sign In

Register Page

Username

Email address

Password

Confirm Password

Sign Up

Wifiology

Error 404: Group Name Not Found

An Unexist Group From CU Boulder

WIFIOLOGY

HOME

HISTORY GRAPH

LIST OF SSID

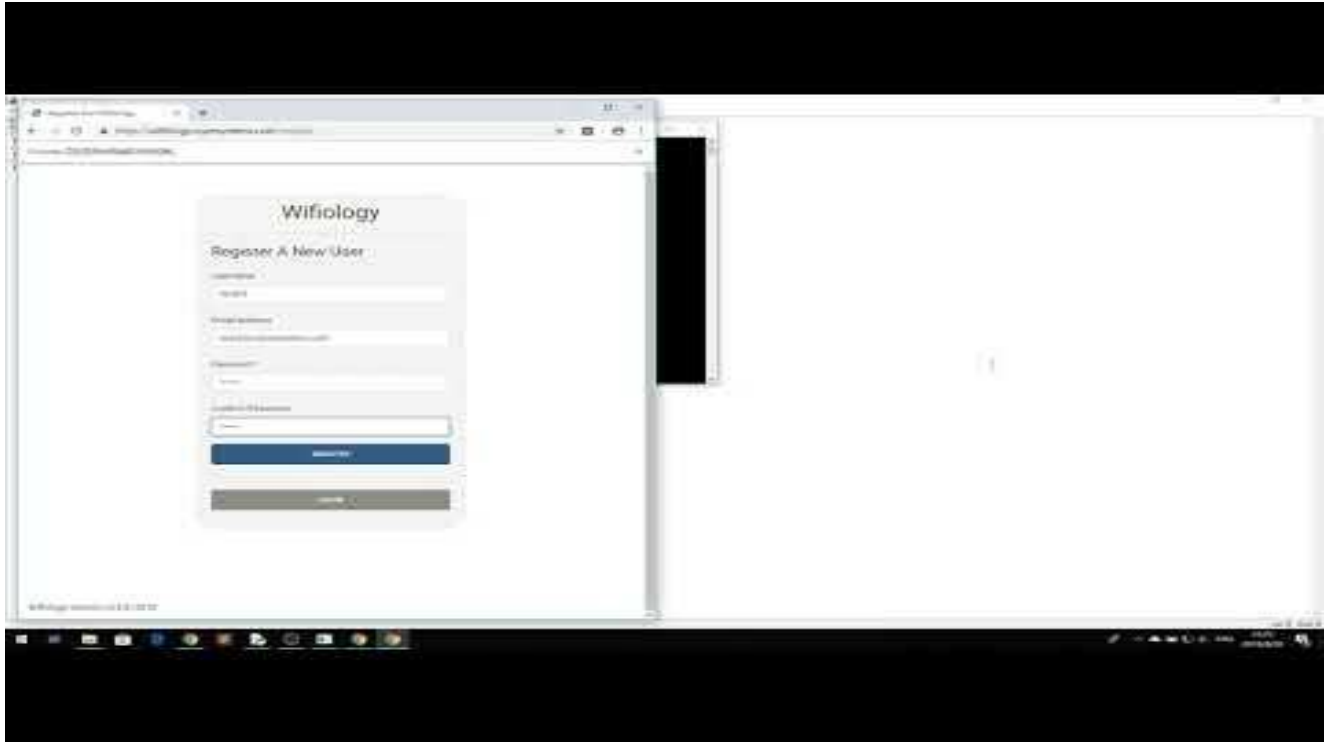
EXPORT DATA

Selenium Automation Testing

- Selenium provide many tools to test the webpage automatically.
- We use selenium to test the register, login and logout function.
- Testing process :
  - Selenium automatically control the web browser to open our website
  - Click the register button and go to the register page
  - Fill the information and click the register button to create a new account
  - Log out
  - Use the new account to login to our website
  - Test done

All testing process are running automatically!

Selenium Automation Testing



Website Demo

https://wifiology.copesystems.com/

Next Steps

- More analysis on the central server
- Easier listener node configuration
- Active probing
- Better multi tenancy
- Easy hardware package for listener node
- Convert listener node to C/C++/??? (Performance)

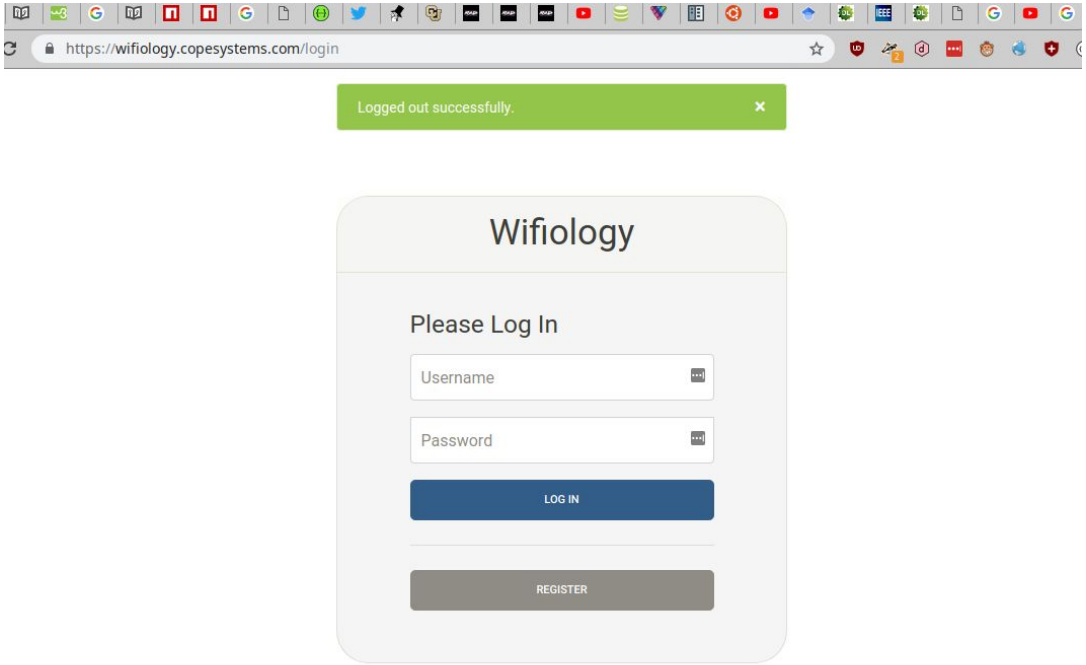
References Used (802.11)

- *“802.11 Wireless Networks: The Definitive Guide, 2nd Ed.”, Gast*
- *“Next Generation Wireless Lans: Throughput, Robustness, and Reliability in 802.11n”, Perahia and Stacey*
- *“Identifying Channel Saturation in Wi-Fi Networks via Passive Monitoring of IEEE 802.11 Beacon Jitter”, Molina, Blac, Montavont, Simić*
- *“Mrc-cciew”* (https://mrncciew.com)

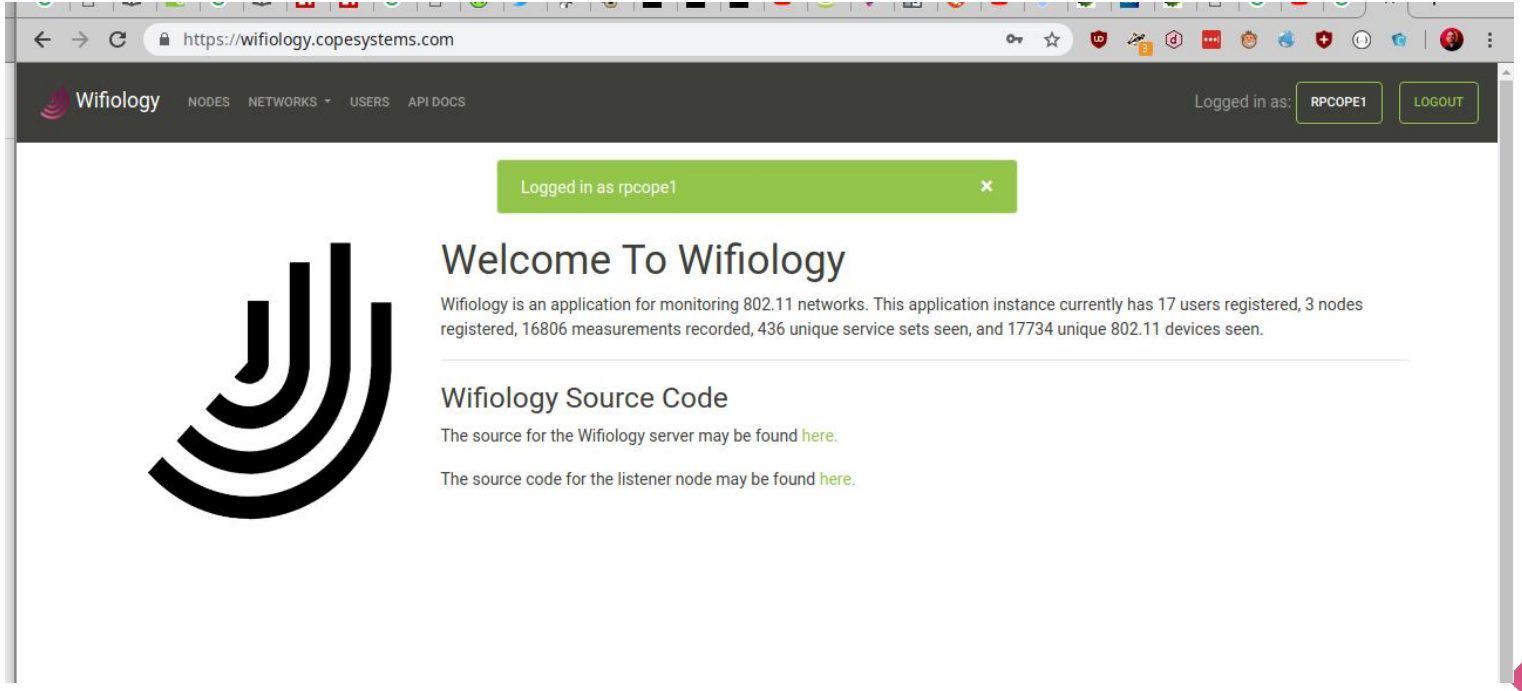
Backup Slides

ONLY IF NEEDED

Website


https://wifiology.copesystems.com/login
Logged out successfully.
Wifiology
Please Log In
Username
Password
LOG IN
REGISTER

Website


https://wifiology.copesystems.com
Wifiology
NODES
NETWORKS
USERS
API DOCS
Logged in as:
RPCOPE1
LOGOUT
Logged in as rpcope1
Welcome To Wifiology
Wifiology is an application for monitoring 802.11 networks. This application instance currently has 17 users registered, 3 nodes registered, 16806 measurements recorded, 436 unique service sets seen, and 17734 unique 802.11 devices seen.
Wifiology Source Code
The source for the Wifiology server may be found here.
The source code for the listener node may be found here.

Website

https://wifiology.copesystems.com/users/me

Wifiology NODES NETWORKS USERS API DOCS Logged in as: RPCOPE1 LOGOUT

User: rpcope1

Username	rpcope1
Email Address	rpcope1@gmail.com
Is Admin	true
Is Active	true
Description	Robert Cope -- Wifiology Admin
User Since:	???

API Keys

Key ID	Key Description	Key Expiry	Delete Key
1	Downstairs Node Key	None	DELETE
2	Laptop Node Key	None	DELETE
3	Raspberry Pi Node Key	None	DELETE
4	Laptop Alt Key	None	DELETE

Create A New API Key

API Key Description (What it will be used for)

Enter description

SUBMIT

Wifiology Version (1.0.1) 2019

Website

https://wifiology.copesystems.com/nodes

Wifiology NODES NETWORKS USERS API DOCS Logged in as: RPCOPE1 LOGOUT

My Nodes

Node ID	Node Name	Node Location	Is Public	Last Seen
1	robertDownstairsNode	Basement	true	Tue Apr 30 2019 23:58:46 GMT-0600 (Mountain Daylight Time)
2	Laptop Node	Roaming	true	Wed May 01 2019 00:01:54 GMT-0600 (Mountain Daylight Time)
3	robertPiNode	Roaming	true	Mon Apr 29 2019 03:46:03 GMT-0600 (Mountain Daylight Time)

Public Nodes

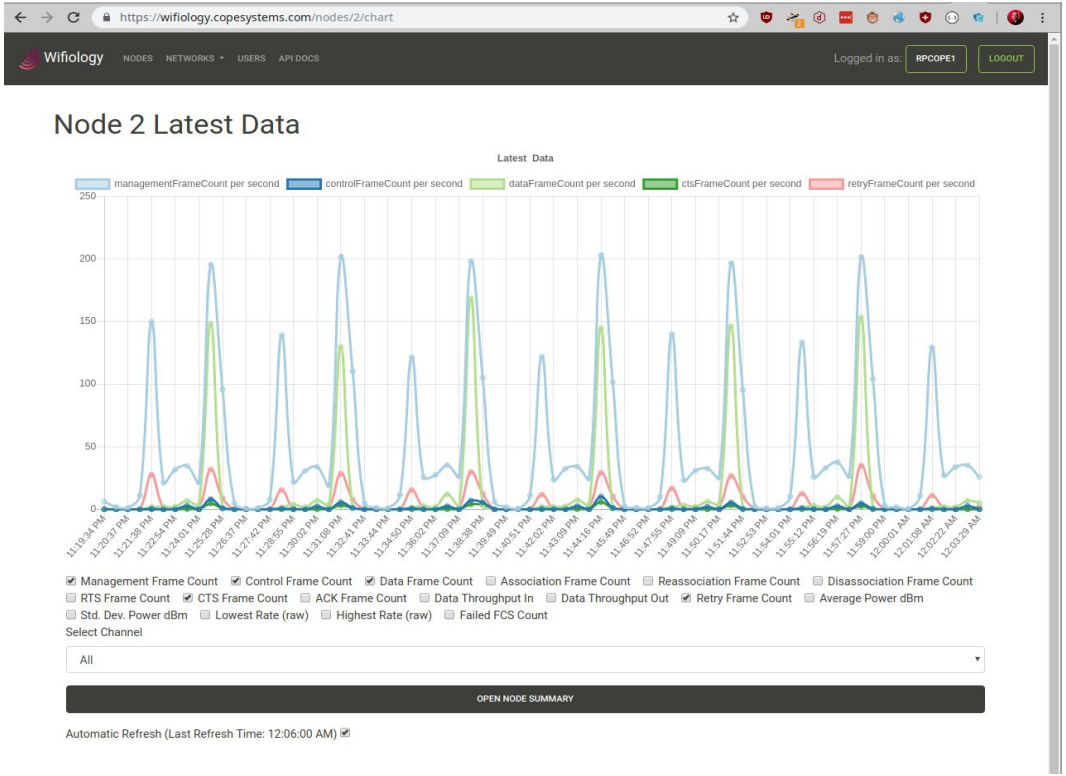
Node ID	Node Name	Node Location	Is Public	Last Seen
1	robertDownstairsNode	Basement	true	Tue Apr 30 2019 23:58:46 GMT-0600 (Mountain Daylight Time)
2	Laptop Node	Roaming	true	Wed May 01 2019 00:01:54 GMT-0600 (Mountain Daylight Time)
3	robertPiNode	Roaming	true	Mon Apr 29 2019 03:46:03 GMT-0600 (Mountain Daylight Time)

REGISTER A NEW NODE

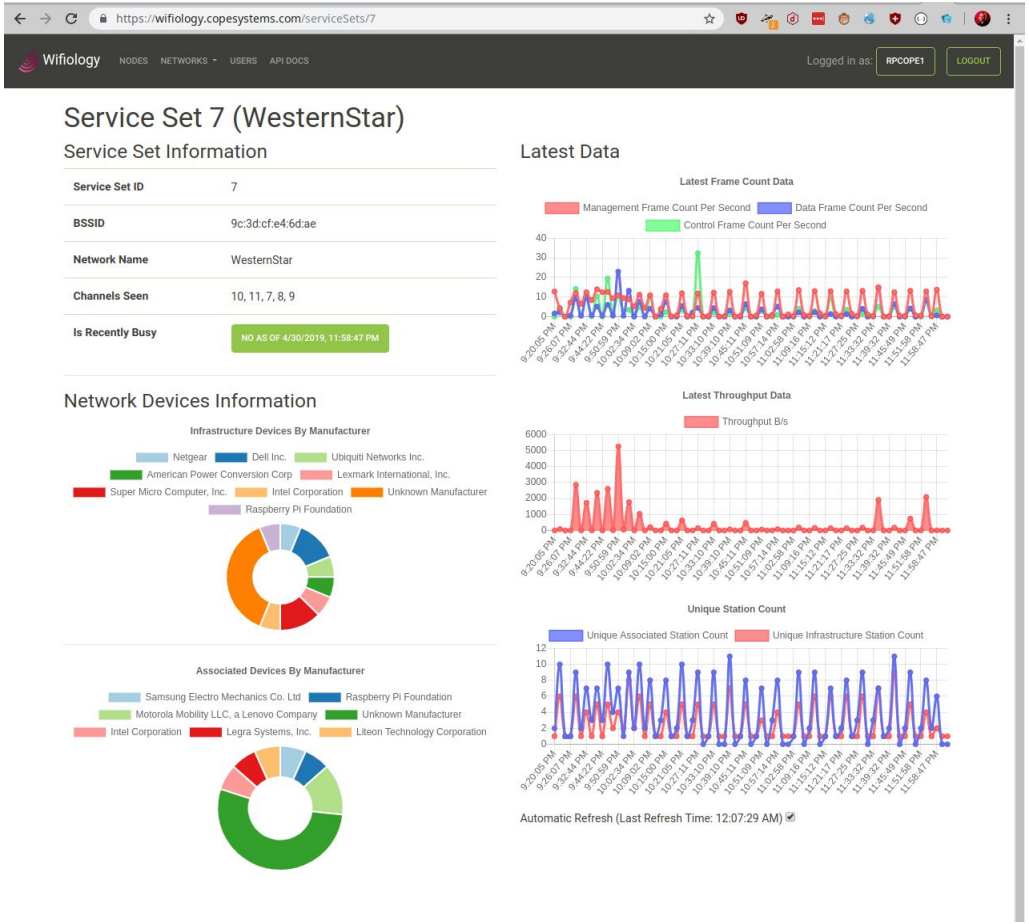
Website


https://wifiology.copesystems.com/nodes/2
Wifiology
NODES
NETWORKS
USERS
API DOCS
Logged in as:
RPCOPE1
LOGOUT
Node Information
Node ID 2
Node Name Laptop Node
Node Location Roaming
Node Description Robert's Laptop
Node Last Seen Wed May 01 2019 00:01:54 GMT-0600 (Mountain Daylight Time)
Area Seems Busy? ???
Recent Service Sets (All Channels)
null (14 Access Points)
Alpine Modern Private (2 Access Points)
(5 Access Points)
CRA Engineering (1 Access Points)
CenturyLink3311 (1 Access Points)
ADITune (1 Access Points)
DIRECT-04-HP OfficeJet 4650 (1 Access Points)
ntap-guest (16 Access Points)
clocks (16 Access Points)
\x00\x00\x00\x00\x00\x00 (8 Access Points)
H+C Guest Network (1 Access Points)
\x00\x00\x00\x00 (1 Access Points)
Bartaco Guest (1 Access Points)
RUConnected (1 Access Points)
JAAP (1 Access Points)
DIRECT-44-HP OfficeJet Pro 8710 (1 Access Points)
Recent Data
Latest Frame Count Data
Management Frame Count Per Second
Data Frame Count Per Second
Control Frame Count Per Second
Latest Throughput Data
Throughput B/s
Unique Station Count
Unique Station Count
Select Channel
Channel 1
OPEN ADVANCED CHART
Automatic Refresh (Last Refresh Time: 12:04:34 AM)

Website


https://wifiology.copesystems.com/nodes/2/chart
Wifiology
NODES
NETWORKS
USERS
API DOCS
Logged in as:
RPCOPE1
LOGOUT
Node 2 Latest Data
Latest Data
managementFrameCount per second
controlFrameCount per second
dataFrameCount per second
ctsFrameCount per second
retryFrameCount per second
250
200
150
100
50
0
Management Frame Count
Control Frame Count
Data Frame Count
Association Frame Count
Reassociation Frame Count
Disassociation Frame Count
RTS Frame Count
CTS Frame Count
ACK Frame Count
Data Throughput In
Data Throughput Out
Retry Frame Count
Average Power dBm
Std. Dev. Power dBm
Lowest Rate (raw)
Highest Rate (raw)
Failed FCS Count
Select Channel
All
OPEN NODE SUMMARY
Automatic Refresh (Last Refresh Time: 12:06:00 AM)

Website


https://wifiology.copesystems.com/serviceSets/7
Wifiology
NODES
NETWORKS
USERS
API DOCS
Logged in as:
RPCOPE1
LOGOUT
Service Set 7 (WesternStar)
Service Set Information
Service Set ID
7
BSSID
9c:3d:cf:e4:6d:ae
Network Name
WesternStar
Channels Seen
10, 11, 7, 8, 9
Is Recently Busy
NO AS OF 4/30/2019, 11:58:47 PM
Network Devices Information
Infrastructure Devices By Manufacturer
Netgear
Dell Inc.
Ubiquiti Networks Inc.
American Power Conversion Corp
Lexmark International, Inc.
Super Micro Computer, Inc.
Intel Corporation
Unknown Manufacturer
Raspberry Pi Foundation
Associated Devices By Manufacturer
Samsung Electro Mechanics Co. Ltd
Raspberry Pi Foundation
Motorola Mobility LLC, a Lenovo Company
Unknown Manufacturer
Intel Corporation
Legra Systems, Inc.
Liteon Technology Corporation
Latest Data
Latest Frame Count Data
Management Frame Count Per Second
Data Frame Count Per Second
Control Frame Count Per Second
Latest Throughput Data
Throughput B/s
Unique Station Count
Unique Associated Station Count
Unique Infrastructure Station Count
Automatic Refresh (Last Refresh Time: 12:07:29 AM)

Swagger Spec

Swagger

Wifiology 1.0.1

[Base URL: wifiology.copesystems.com/api/1.0]

/api/1.0/api-docs

Authorize

API Keys

- GET /users/me/apiKeys Return a users keys.
- POST /users/me/apiKeys Register a API key.
- GET /users/me/apiKeys/{apiKeyID} Return an individual key.
- DELETE /users/me/apiKeys/{apiKeyID} Delete an API key.

Measurements

- GET /nodes/{nodeID}/measurements Get the latest measurements and information
- POST /nodes/{nodeID}/measurements Load a new measurement.

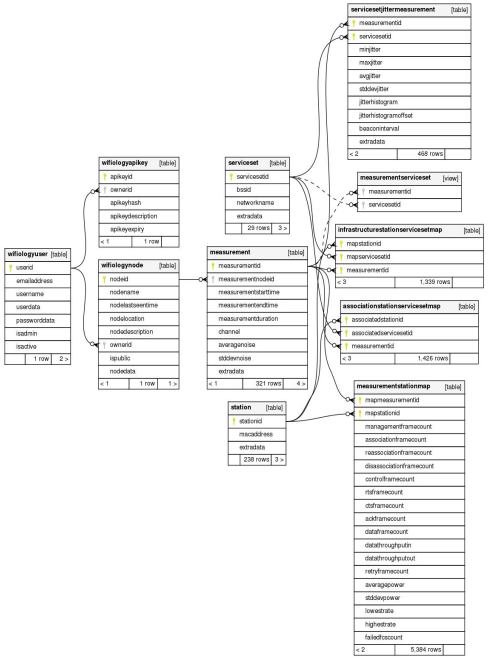
Nodes

- GET /users/me/nodes Return the current users nodes.
- GET /nodes Return a list of all visible nodes.
- POST /nodes Register a new node.
- GET /users/{userID}/nodes Return the visible nodes for a given user.
- GET /nodes/{nodeID}/measurements Get the latest measurements and information
- POST /nodes/{nodeID}/measurements Load a new measurement.

Test Stuff

- GET /testThing Returns worlds by name

Full PostgreSQL Schema (Huge)



featureflag	[table]
featureflagkey	varchar[512]
featureflagvalue	jsonb[2147483647]
< 0 0 rows	0 >

migrations	[table]
id	serial[10]
name	varchar[255]
run_on	timestamp[29,6]
< 0 5 rows	0 >

Python

```
rcope@mentlegen:~/git/client_proof_of_concept$ cat requirements.txt
pcapy==0.11.4
pytimerfd
signalfd
dpkt
pyric
bottle
assertpy
manuf
eventlet
nose
requests
scapy
hdrhistogram
rcope@mentlegen:~/git/client_proof_of_concept$
```

NodeJS

```
{
  "name": "Wifiology",
  "version": "1.0.1",
  "description": "Getting the basics set up for Wifiology",
  "engines": {
    "node": "10.x"
  },
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "test": "mocha --recursive test/*",
    "migrate": "db-migrate",
    "janitor": "node janitor.js",
    "pm2": "pm2",
    "pm2_start": "pm2 start server.js"
  },
  "dependencies": {
    "ejs": "^2.5.6",
    "express": "^4.15.2",
    "pg": "^7.8.2",
    "express-openapi": "^4.5.0",
    "swagger-ui-express": "^4.0.2",
    "node-postgres-named": "^2.4.1",
    "openapi-security-handler": "^2.0.4",
    "basic-auth": "^2.0.1",
    "body-parser": "^1.18.3",
    "winston": "^3.2.1",
    "express-winston": "^3.1.0",
    "axios": "^0.18.0",
    "db-migrate": "^0.11.5",
    "pm2": "^3.5.0",
    "db-migrate-pg": "^0.5.0",
    "sync": "^0.2.5",
    "passport": "^0.4.0",
    "passport-local": "^1.0.0",
    "express-session": "^1.16.1",
    "cookie-parser": "^1.4.4",
    "connect-flash": "^0.1.1",
    "email-validator": "^2.0.4",
    "hdr-histogram-js": "^1.1.4",
    "connect-pg-simple": "^5.0.0",
    "oui": "^9.1.11"
  },
  "devDependencies": {
    "chai": "^4.2.0",
    "mocha": "^6.1.4",
    "random-mac": "^0.0.5",
    "request": "^2.81.0",
    "request-promise-native": "^1.0.7",
    "tape": "^4.7.0"
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/404-group-does-not-exist/Wifiology"
  },
  "keywords": [
    "node",
    "heroku",
    "express"
  ],
  "license": "MIT"
}
```

Node.js is Rock Star Tech