Milestone 4 - Wifiology Design

CU CSCI 3308

March 17 2018

404-Group-Not-Found

Jason Nguyen, Robert Cope, Baiyu Chen, Ryan Campbell, Peng Jiang, Jasper Niemeyer

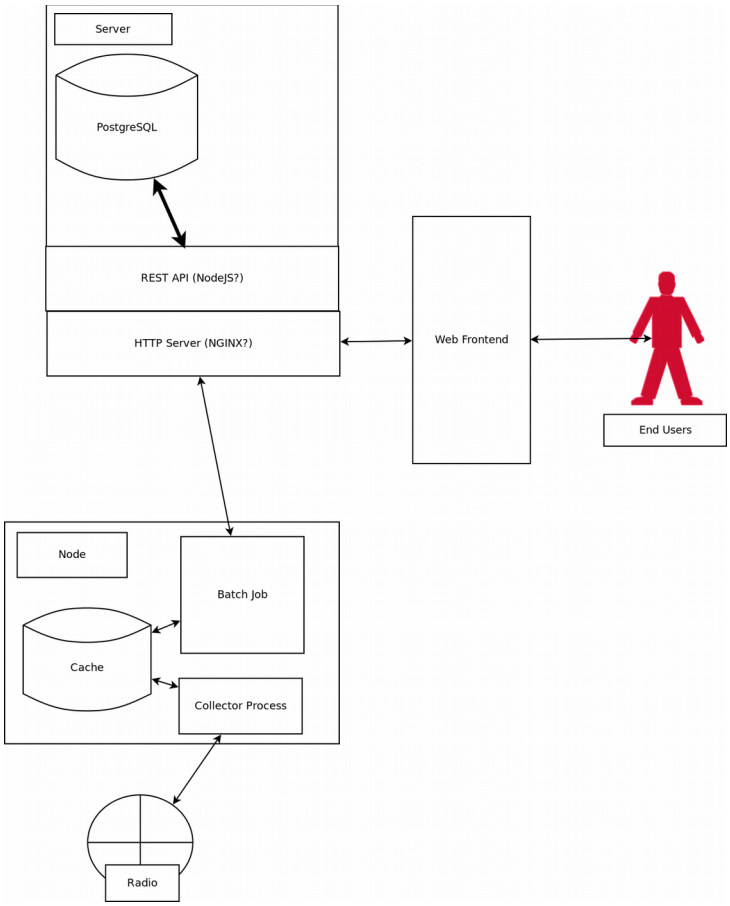
Purpose

Wifiology is a service which informs users and administrators about wireless usage and congestion at their favorite hot spots, helps users schedule access time to avoid highly congested networks, and helps administrators see metrics on how their 802.11 networks are performing.

High Level Design

Wifiology uses a hub and spokes design, with a central server and many nodes communicating with the central server. The central server provides both a traditional web user interface, for 802.11 consumers and admins to connect to, get data and analyze data, as well as a REST API which is used both for the nodes to communicate to the server with, as well as allowing users programmatic acess to the data available from our service. The nodes are physical systems co-located with the 802.11 networks under analysis. These nodes use a special 802.11 card to capture traffic, and send it to the central server in batches as the data is ready.

Architecture Diagram



Server Design

The server for Wifiology serves two purposes: (1) to communicate the data captured and analyzed by the service via a traditional web interface (with additional REST API access if the users require it), and (2) as a way for Wifiology nodes to communicate their capture data to a central database via a REST API.

The server will be written with the latest LTS version of Node.JS and the express.js server framework, and will use PostgreSQL as our backing database.

Front End Design

The application will utilize a traditional web front end, rendered server-side (as opposed to a single page application (SPA) style web application). We plan to utilize Bootstrap 4 in conjunction with our own customizations, which will be rendered with ejs and served by express.js. Users will be required to log in to view any of the data, manage their nodes, or alter any configuration. Authentication will utilize one of the following: HTTP basic auth, JSON web tokens (JWT) or OAuth2 (depending on further design considerations).

Each node will have a page to display its historical traffic data.

Wire Frame Mock-Ups of Web Design

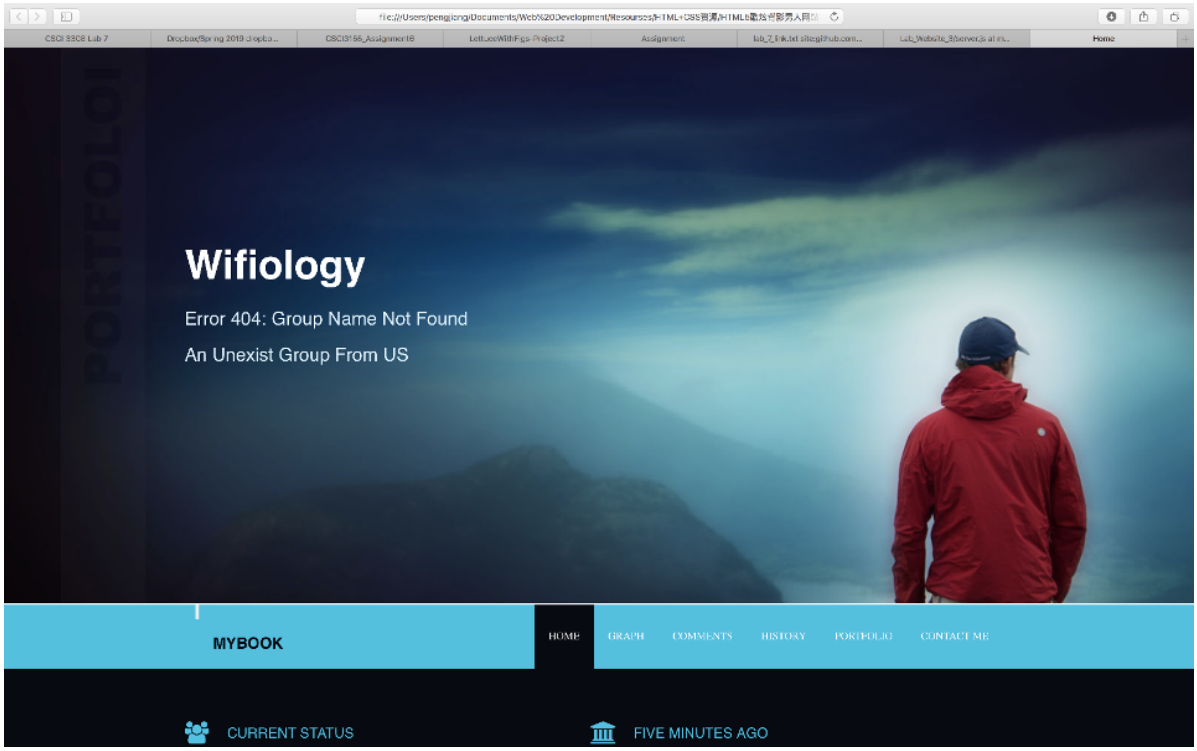


Illustration 1: Landing Page

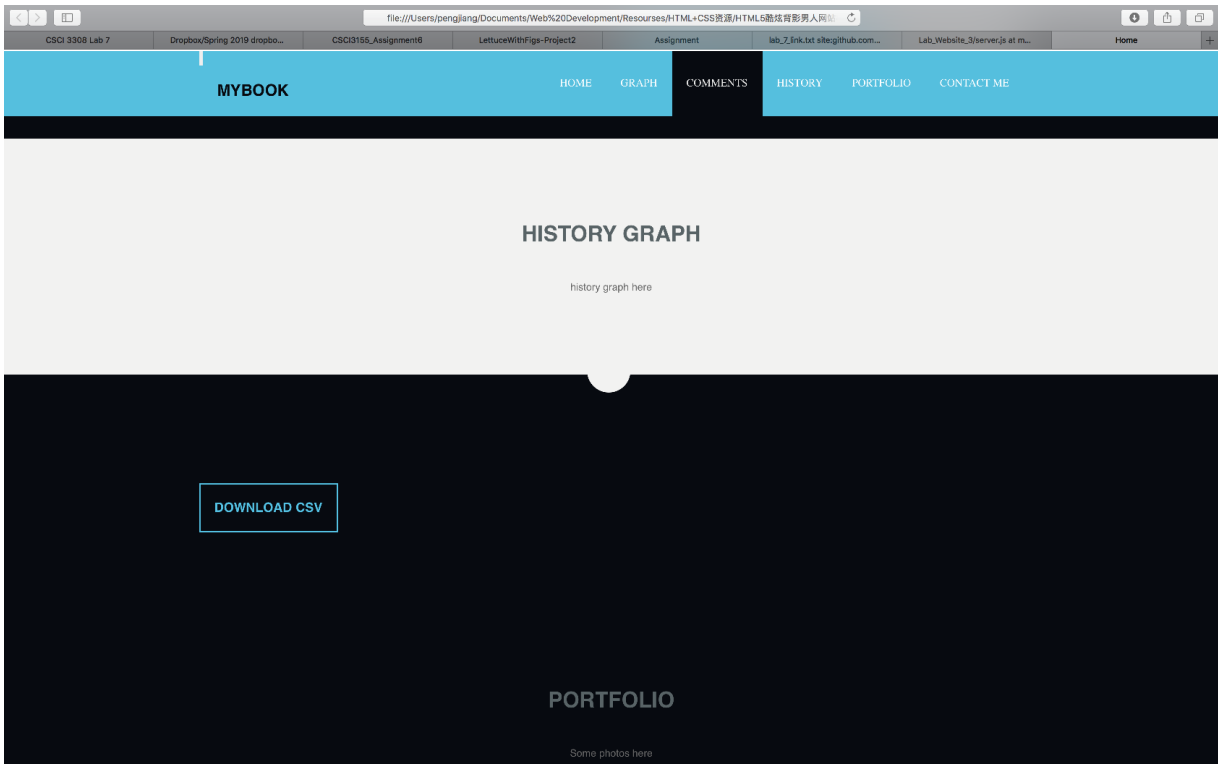


Illustration 2: Page to display graph of data and download csv

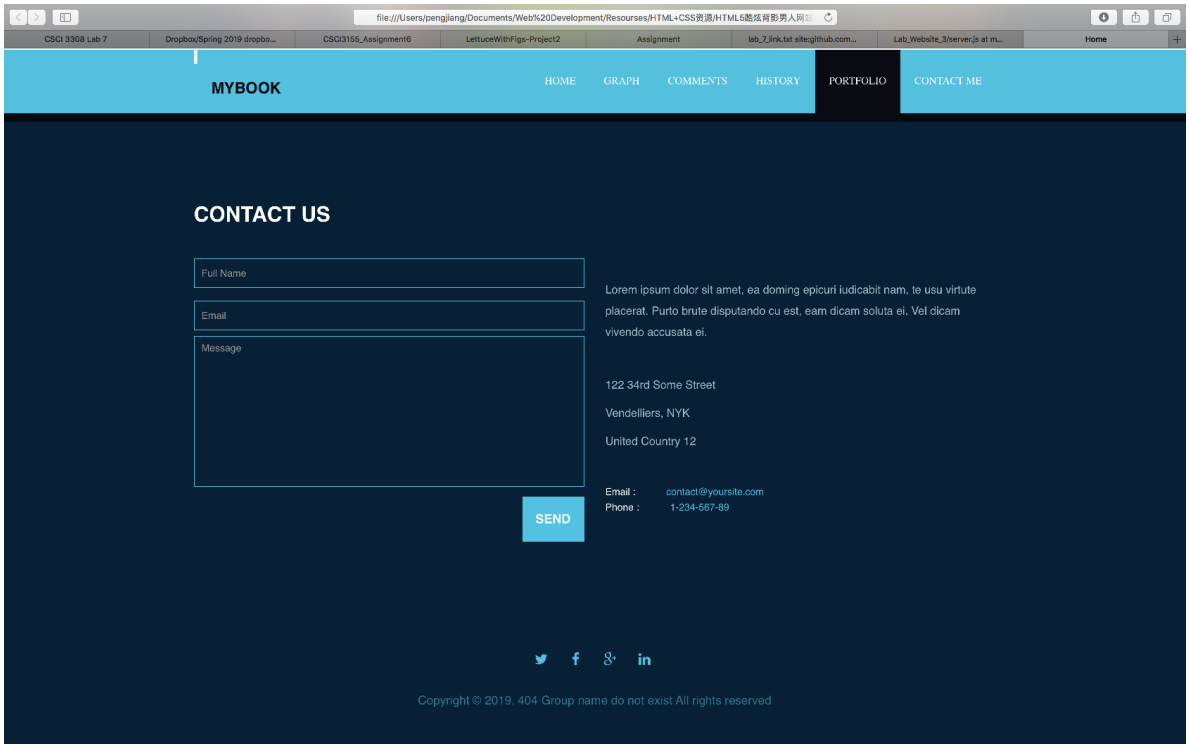


Illustration 3: Contact page

Web Service Design

This service is not currently designed to integrate with any third party web services.

Revised List of Features

At this time our planned list of features remains the same as it was for Milestone 2.

1. check traffic on node

2. list out available wifi connections

3. list out traffic on each connection

4. export data to CSV (for user use and for admin use)

5. view time stamp of last batch update to DB

6. Provide users with notifications

Node Design

The data for Wifiology will be captured by "nodes" (small low power ARM, MIPs or x86 systems colocated in the physical area of interest for 802.11 traffic), which are constantly capturing samples of 802.11 traffic, analyzing it, and uploading in batch to the Wifiology server. The nodes will each have unique IDs known a priori which will allow the Wifiology server to give users information about 802.11 usage and congestion.

Each of the nodes will have one 802.11 capture card capable of utilizing "monitor mode" (which enables capture of all 802.11 traffic on a given channel) and compatible with the Linux mac80211 kernel module. Additionally either a cellular modem, an ethernet connection, or another 802.11 card connected to an access point with internet access is required; this device is necessary for the node to upload the aggregated capture data, but is not necessary to be always on or completely reliable.

Each of the nodes will be running a modern Linux distribution (where modern is defined to be a distribution release within the last 2-3 years, and a kernel version that is 3.*.*+ or newer). The nodes will leverage the mac80211 kernel module, with the libpcap library, the iwtools toolkit and library, and a RadioTap decoder in conjunction with custom Python code (with bindings to the above) to capture, decode, and analyze the captured traffic. Once preliminary analysis and aggregation is completed on the capture data, the data is stored in an (optionally in-memory) SQLite3 database.

At scheduled intervals, an upload script is run on the node. The upload script opens the SQLite3 database above, and queries for new data since the last time it was run. If new data is found, the upload script will attempt to upload the node data to the Wifiology server via a REST API. If the upload is successful, the data is (optionally) purged from the local database, and further clean-up is performed. If the upload is unsuccessful, the script aborts until the next scheduled upload time, and the database in the local database is left unmodified. The scheduling and execution of the upload script will be driven

by cron. We expect it to be run anywhere from every 3-5 minutes if the uplink has no bandwidth constraints, to every 30 minutes to an hour if the uplink to the server is bandwidth constrained.

Capture and Analysis Algortihm (High Level)

The first step of the capture and analysis algorithm is to ensure that the capture card is in monitor mode. This is performed by using the iwtools library to query the card state, and adjust it to monitor mode. Other basic sanity checks on the cards state will be performed at this time. Once the card is ready, we iterate through each 802.11 channel (1 to 11). We set the card to listen on the current channel, and begin capture with pcap. Because the analysis is too slow to keep up with capture, while capture is occurring, the data is written to a pcap file on a tmpfs mount (i.e. the capture data is kept in memory for performance). The capture runs for a known, set amount of time (typically 30 seconds). Once the capture time is done, the pcap capture interface is closed.

After the capture phase for this channel, the capture file is reopened with Pcap, and analysis is performed offline. The packets received are sorted by time received, and binned into their corresponding 802.11 frame types. The analysis passes over the data multiple times. The first time, all of the beacon frames are found, and are used to correlate SSIDs (such as "CU Wireless") with all of the base station MAC addresses advertising that they provide that SSID. The SSIDs seen for the channel, as well as their AP radio stations (with their MAC address(es) and relative power) are saved.

A second pass of the data is then performed, and all of the data and control packets are analyzed. The data packets are aggregated to get a relative measure for each SSID and each base station how much data traffic is being both sent and received, and how many devices were seen communicating during the capture window. The control packets are analyzed to see how many "request to send" versus "cleared to send" packets where seen, and in conjunction with other to-be-defined metrics derived from the control packets, the a relative conjestion level is derived from these.

All of these aggregate measures are then converted into a data structure amenable to storage in a relational database, and commited into the local SQLite database. With the analysis done for this channel, the capture script checks the status of the capture card, changes to the next capture channel, and the process begins again.

Background Reading on Wireless Capture and Analysis

Books

"Next Generation Wireless LANs: 802.11n and 802.11ac", Perahia and Stacey, Cambridge University Press; 2nd edition (June 24, 2013)

"CWSP Certified Wireless Security Professional Study Guide: Exam CWSP-205, 2nd Edition", Coleman, Sybex; 2nd edition (September 26, 2016)

Papers

"Understanding Congestion in IEEE 802.11b Wirless Networks"; Jardosh, Ramachandran, Almeroth, Belding-Royer; Usenix IMC 2005, (https://www.usenix.org/legacy/event/imc05/tech/full_papers/jardosh/jardosh_new.pdf)

"Performance Analysis of real-time traffic over 802.11n Wireless Local Area Networks: An Experimental Study"; Podolanko, Datta, Das; IEEE 2014 International Wireless Conference Proceedings (https://ieeexplore.ieee.org/document/6906399)

"Congestion Analysis of IEEE 802.11 Wireless Infrastructure Local Area Networks"; Kaur; Global Journal of Computer Science and Technology Vol. 9 Issue 5 (2010) (https://computerresearch.org/index.php/computer/article/download/869/868)