2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) | 978-1-6654-0690-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/UEMCON53757.2021.9666697

DenCity: A WiFi Location Tracking Solution

Joseph Spanilo
School of Computing and Data Science
Wentworth Institute of Technology
Boston, MA, USA
spaniloj@wit.edu

David Edwards
School of Computing and Data Science
Wentworth Institute of Technology
Boston, MA, USA
edwardsd1@wit.edu

Sunjae Park
School of Computing and Data Science
Wentworth Institute of Technology
Boston, MA, USA
parks6@wit.edu

Mira Yun
School of Computing and Data Science
Wentworth Institute of Technology
Boston, MA, USA
yunm@wit.edu

***Abstract*—Google Maps has made it easy for people to plan their commute based on how busy a road is. This has also been expanded to help users understand how busy a store may be during the day. However this feature has not been applied to buildings with much depth. *DenCity* aims to solve this problem by showing a way that the number of users can be tracked without having to implement new hardware to existing infrastructure and without the user having to participate. *DenCity* uses the WiFi signals sent out by devices to track the number of users near an access point. As a result it makes it possible for users to know the number of people within a building or given area.**

***Index Terms*—wireless, mobile and wireless IP, location management,**

I. Introduction

Google Maps [1] has made it easy for people to plan their commute based on how busy a road is. This has also been expanded to help users understand how busy a store may be during the day. If you pull up a business or public space on google you can clearly see how busy the space is by the hour under the ”Popular Times” section [2]. However, this feature does not give you raw numbers as to how many people are actually there. Instead, it only shows you a rough graph showing the increase or decrease. You cannot tell if there are typically 10 people or 100 people at any given time.

This feature has also not been applied to buildings with much depth. Current systems and solutions are not providing which section of a building, or otherwise small area, is more busy than others. This can be especially relevant at amusement parks, museums, hospitals, super stores, and more. Having this information can be convenient for consumers and visitors to plan their shopping and trips [3]. It can also be life saving for buildings to plan for emergencies [4], for hospitals to optimize their layouts and prevent bottlenecks [5], or for tracking disease spread during a pandemic [6].

DenCity aims to solve this problem by showing a way that the number of users can be tracked without having to implement new hardware to existing wireless local area networks and without the user having to participate. Instead of using dedicated wireless sensor networks [7], *DenCity* uses signals sent out by existing access points to track the number of users near an access point.

The WiFi client devices such as laptops, tablets, and mobile phones at each access point are reconfigured to run in promiscuous mode, which allows the WiFi devices to listen to all nearby wireless traffic. *DenCity* process information using the Kismet framework and identify how many devices are connected to each router. This information is then displayed on a website. Users can use this website to know the number of people within a building or given area, and plan accordingly. The rest of this paper is organized as follows. In Section II, we present our design and implementation of *DenCity* in detail. In Section III, we share our findings from our testing results. Finally, Section IV presents our conclusions and final thoughts.

II. DenCity

DenCity aims to create a WiFi tracking solution that requires minimal user interaction and setup. As a result we decided that it must collect all raw data without any user input and that it should be deployable on off-the-shelf access points or low cost single board computers. Then our solution can be cost effective and easy to implement for non-technical consumers. In order to fulfill these requirements we decided to deploy three Raspberry Pi’s as our testing access points. This is because of their availability, cost effectiveness, small size, and ease of use.

For our physical data collection we decided to use Kismet [8]. Kismet is a wireless network and device detector, sniffer, wardriving tool, and wireless intrusion detection system (WIDS) framework. We used Kismet to collect the wireless signal strength of all devices that were connected and to determine whether the device was an access point or a user device. After collecting the data it is filtered and converted into a text file and sent to an Amazon Web Services

(AWS) server. This is then read by a web application written in React [9] and displayed to the user.

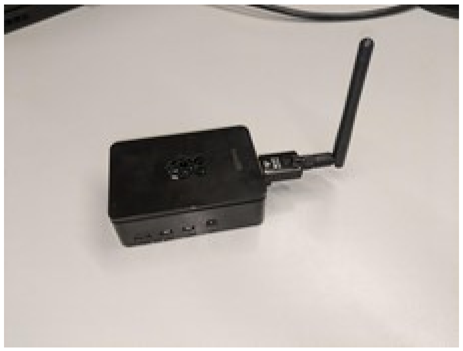


Fig. 1. Raspberry Pi 4 with WiFi Nation WN-H1 adapter

A. *Configuration*

For our data and signal collecting we utilized two Raspberry Pi 4s and one Raspberry Pi 3B+ both running Kali Linux OS. We chose Raspberry Pi's because they are small enough for easy deployments and they only require a functioning power outlet to collect raw data once configured. An example of our setup can be seen in Figure 1. Each Raspberry Pi was connected with a WiFi Nation WN-H1 network adapter that allowed for signal collection. This adapter was running the RTL88AU chipset which is one of the few chipsets left for Wi-Fi adapters that allows promiscuous mode configuration and is compatible with the Kismet software [10]. Promiscuous mode allows the WiFi devices to process all packets it sees, instead of only packets that are destined for the WiFi device [11].

B. *Kismet*

Kismet is one of the most popular wireless network and device detector, sniffer, and WIDS framework [8]. In this application we utilized the device detector portion of this framework to gather each unique WiFi ping from devices in an area. Kismet however collects a lot of bulk and extra data we do not need into the SQLite database, including signal strength and packet information. This caused our data outputs to be extremely cluttered and large. To break down this data we utilized a combination of SQL queries, grep and word count (wc) utilities to get the information we needed.

```
sqlite3 -csv Kismet*.kismet \\
    "SELECT strongest_signal FROM devices
    WHERE type != 'Wi-Fi AP'" \\
    > Kismetfile.csv
```

Listing 1. SQL query used

The query used is shown in Listing 1. This query takes the SQLite file, converts into a csv file named Kismetfile.csv. It selects the strongest signal column from the devices table and checks to see if the device type is 'Wi-Fi AP'. If it is then it will omit it from the output. If the device type does *not* it will populate it in the output file.

The CSV file is then processed to remove all 0s and count the number of lines as shown in Listing 2.

```
grep -i -v '0' Kismetfile.csv | wc -l >
    Kismetfile.txt
```

Listing 2. Grep and WC command to refine data

The file first piped through grep to remove all lines containing zero from the file. The reason for this is because when kismet produces a strongest signal with zero it is either a duplicate ping or it's an error in the data collection for that device. For that reason we decided the best course of action was to remove the lines containing zero since it would remove duplicates and possible dead unimportant data points. It is then processed by the 'wc' command to count the number of lines left in the file after grep removes all the 0s. This is then saved into another file which is sent to the AWS web server. This was done since each line represents a device so by simply counting the lines we can get the number of devices that were a part of the dataset.

The naming convention for our files was based on the location. For example the Douglass D. Schuman Library was named 'KismetFileL.txt', Beatty Hall was 'KismetFileB.txt' and Wentworth Hall 'KismetFileW.txt', making it easy to determine which file was associated with what location.

After organizing and getting our data into a usable format we needed to send our information to our AWS web server. This was done by using the Secure Copy Protocol (SCP); this option was one of the cleanest and most efficient ways available while also having the benefit of being secure. For this application we did not need all the features that Secure File Transfer Protocol (SFTP) can offer and just needed to send files from host to host which SCP does perfectly.

This was then tied into a bash script which ran all of these commands from starting Kismet all the way to sending the data to the web server. This script was set to run every 5 minutes on each web server using a cron job. Cron is a time-based job scheduler in Unix or Unix-like computer operating systems. The body of the script can be found in Listing 3.

The Kismet line is ran using the -c flag and the collection source which is wlan0 (the RTL88AU chipset adapter). The ID of the Kismet process is saved in the 'PID' variable. After 30 seconds it kills the Kismet process. It then runs through both the other code blocks in Listing 1 and Listing 2. After that it sends the file to our AWS web server using SCP with our AWS Privacy Enhanced Mail (PEM) key. After the file is sent all the files auto-generated from the capture are then deleted and cleared for the next one.

```
#!/bin/bash
kismet -c wlan0 &
PID=$!

# Briefly sleep to wait for kismet,
# then stop it using the kill command
sleep 30s
kill -INT $PID
sleep 10s
```

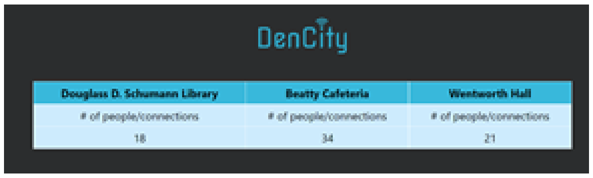
```
# Take db get strongest signal/rm WiFi APs and
# convert to csv
sqlite3 -csv Kismet*.kismet \\
    "SELECT strongest_signal FROM devices
    WHERE type != 'Wi-Fi AP'" \\
        > Kismetfile.csv
sleep 5s

# Remove all 0s and count number of lines
grep -i -v '0' Kismetfile.csv | wc -l >
    KismetFileL.txt
sleep 5s

# Copy files to AWS Webserver
scp -i ~/Desktop/AWS_Kismet_EC2_ED.pem
    KismetFileL.txt \\
    ubuntu@aws_webserver:~/Desktop/
sleep 5s
echo "done"

# Remove generated Kismetfile
rm Kismet-*
```

Listing 3. Full automation script

C. *Web Application Design and Implementation*

In order to implement our solution we decided to create a web application using JavaScript, HTML, and CSS. As shown in Figure 2 , we chose to use React for the user interface (UI) because of its ease of use when it comes to implementing a web application that updates with the information given.

We utilized Node.js for our backend. The data is read from the txt file that was gathered by the Raspberry Pi's. The data is displayed on the web application using React. This updates in real time for the user as the user views the program.



Douglass D. Schumann Library	Beatty Cafeteria	Wentworth Hall
# of people/connections	# of people/connections	# of people/connections
18	34	21

Fig. 2. DenCity UI using React

D. *AWS Deployment*

For our web server deployment a AWS EC2 was used running Ubuntu with XAMPP which is a pre-packaged Apache web server distribution. The EC2 instance was configured as a t2.micro instance because it has the amount of processing power needed to host our web application. Additionally, since all of the data parsing and organizing through file is handled by the Pi's, the webserver does not have to handle any of the processing further reducing the need for a more powerful server.

In order to Secure Shell (SSH) into our webserver as well as SCP our files to it, we needed to use the .pem file. This is essentially an encrypted key that is generated and used by the EC2 instance to authenticate SSH connections and requests. Each EC2 instance has its own unique .pem file.

III. Evaluation

When conducting tests for *DenCity* we recorded how many users were in the area at a given time and then compared that to how many connections we received from our output of *DenCity*. Below is our data collected from Douglas D. Schuman library at Wentworth Institute of Technology in 10 tests across 10 different days in Table I.

TABLE I
Data collection results

Test	Actual	Application
1	20	19
2	15	16
3	21	21
4	25	24
5	13	14
6	24	22
7	16	15
8	18	12
9	17	17
10	23	23

From these tests we found that our data collection was consistent and always being within 1 to 2 persons of what we actually counted, except for one outlier on test 8. This meant our application was collecting data accurately and from these tests had 4.6875% of error rate.

IV. Conclusion and Future Work

In *DenCity* we were able to show that it is possible to reuse existing access points to collect users location. There are however ways that we can refine and streamline *DenCity*. First would be to implement a map system on the home page. This would allow the users to view the locations that are being tracked and see a visual representation of the number of people in a given area. Second would be to implement trilateration to our program. This would allow *DenCity* to have more accurate tracking that is better suited for room by room tracking. Furthermore improvements to the UI should also be made to ensure a better user experience. Another major feature would be the ability to view tracking records within the web application easily. This would make *DenCity* much more suited for analytics and significantly increase its utility.

Overall this *DenCity* project has laid the foundation for what could become a useful tool for location tracking. *DenCity* is reasonably accurate and cheap for business and institutions to implement. Potentially it could be implemented with existing tools such as Google Maps to maximize its utility and accessibility.

References

[1] Google Maps. [Online]. Available: https://maps.google.com/. [Accessed: 02-Jun-2021].

[2] Google Maps will soon indicate how busy a place is directly on the map. [Online]. Available: https://9to5google.com/2020/10/15/google-maps-indicate-busy-directly-on-map/ [Accessed: 15-Oct-2021].

[3] Didi Surian, Vitaliy Kim, Ranjeeta Menon, Adam G. Dunn, Vitali Sintchenko, Enrico Coiera, ”Tracking a moving user in indoor environments using Bluetooth low energy beacons,” Journal of Biomedical Informatics, https://doi.org/10.1016/j.jbi.2019.103288.

[4] A. Desmet and E. Gelenbe, ”Reactive and proactive congestion management for emergency building evacuation,” 38th Annual IEEE Conference on Local Computer Networks, 2013, pp. 727-730, doi: 10.1109/LCN.2013.6761321.

[5] Tudor Morar, Luca Bertolini, ”Planning for Pedestrians: A Way Out of Traffic Congestion,” Procedia - Social and Behavioral Sciences, Volume 81, 2013, Pages 600-608, ISSN 1877-0428, doit: https://doi.org/10.1016/j.sbspro.2013.06.483.

[6] Van Romero, William D. Stone, Julie Dyke Ford, ”COVID-19 indoor exposure levels: An analysis of foot traffic scenarios within an academic building,” Transportation Research Interdisciplinary Perspectives, Volume 7, 2020, doit: https://doi.org/10.1016/j.trip.2020.100185.

[7] S. Faye and C. Chaudet, ”Characterizing the Topology of an Urban Wireless Sensor Network for Road Traffic Management,” in IEEE Transactions on Vehicular Technology, vol. 65, no. 7, pp. 5720-5725, July 2016, doi: 10.1109/TVT.2015.2465811.

[8] Kismet. [Online]. Available: https://www.kismetwireless.net/. [Accessed: 02-Jun-2021].

[9] React. [Online]. Available: https://reactjs.org/ [Accessed: 02-Jun-2021].

[10] “Hardware,” Kismet. [Online]. Available: https://www.kismetwireless.net/hardware/. [Accessed: 08-Jun-2021].

[11] Ansari, Sabeel, S. G. Rajeev, and H. S. Chandrashekar. ”Packet sniffing: a brief introduction.” IEEE potentials 21.5 (2003): 17-19.,