UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Geio Illus

Wi-Fi Positioning System

Bachelor's Thesis (9 ECTS)

Supervisor: Danielle Morgan, Msc

Tartu 2021

Wi-Fi positioning system

Abstract:
The result of this thesis is a database with Wi-Fi routers located in the Delta Building and the strength of the signal of these routers at various locations in the building. To collect the necessary data about access points, a program was developed. In addition, an additional proof of concept script was also developed to locate a person based on the collected data. The program is written in Python and uses a MySQL database to hold the data.
Keywords:
Wi-Fi, access points, WLAN, mapping, Python, MySQL

CERCS:

P170 Computer science, numerical analysis, systems, control

P175 Informatics, systems theory

Wi-Fi positsioneerimise süsteem

Lühikokkuvõte:
Käesoleva töö tulemusel on loodud andmebaas Wi-Fi ruuterite ja nende signaalide tugevustega maja erinevates osades Delta hoones. Andmete kogumiseks on loodud programm, et ligipääsupunkte kaardistada. Lisaks kaardistamisele on kontseptsiooni tõestamiseks loodud ka skript, mis üritab kindlaks määrata inimese asukohta antud hoones. Programm on kirjutatud kasutades Python programmeerimiskeelt ning MySQL andmebaasi, kuhu salvestatakse andmed.
Võtmesõnad:
Wi-Fi, ligipääsupunktid, WLAN, kaardistamine, Python, MySQL

CERCS:

P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

P175 Informaatika, süsteemiteooria

Table of Contents

1 Introduction

People have managed to successfully travel between cities, establish international trade routes by land and explore the world by sea for thousands of years already. This has been achieved thanks to the earliest developments of cartography and the art of navigation. For example, the earliest maps that have survived have been etched on tusk and stone [1]. Seafaring was usually done by keeping land in their view at all times as to not get lost. Sometimes this method could not be used however as there were times when boats had to sail further away from the land. The most common way of wayfinding in that case was celestial navigation or by searching for landmarks. The sailors even used waves and cloud patterns to determine the presence of the land [2]. As time progressed, more wayfinding solutions were developed. The first compass for navigation was used in China about a thousand years ago [3]. A few hundred years later came a sextant, making determining the longitude fairly accurate. More modern solutions include sending out radio signals, radars and using Global Positioning System (GPS).

The GPS project was launched in 1973. The first GPS satellite was launched in 1978 and its initial intended use was for navigation by the United States military. Civilian use was allowed from the 1980s. However, at first, the highest-quality signal was reserved for military use and intentionally degraded for civilian use. This policy is known as Selective Availability. On May 2, 2000 this policy was turned off, granting the civilians the same accuracy as it did for the military. [4]

There have been multiple improvements to this system throughout the years since then. Due to the GPS service belonging to the United States government, they can selectively deny access to the system as well [5]. As a result, some countries have been developing other satellite navigation systems. The most notable currently being GLONASS which is created by Russia and Galileo, which is created by European Union. The latter being able to determine the position with an accuracy of less than 1 metre [6].

As good as these systems are, they can not accurately tell what floor the person is on or in what room exactly if they are in a building with multiple floors and rooms. The purpose of this thesis is to provide a way for people to navigate around in the Delta building in the future. Due to this, another way to determine the user's location was needed. This can be done by scanning for wireless access points in the building and using the received signal strengths from the data emitted by them, for example [7]. However, for this to work correctly, data about the signal strength is needed to be collected first. Although there are numerous Wi-Fi and network analysers available that can detect and show the user the nearby networks and their signal strengths, essentially none of them can be used to save the necessary data for future use without the need of processing that data beforehand. Therefore, a custom solution had to be developed to extract and save the useful information about the access points. To achieve this, a program was written using Python and data was saved to a MySQL database which can be used as a groundwork for future works.

The thesis is organized as follows: Chapter 2 gives a detailed overview of used technologies and explains the terminology used for this thesis. Chapter 3 describes the program architecture and design. Chapter 4 gives an overview of the results and also talks about the arisen problems and limitations. Finally, offers new opportunities to continue to expand the project and make it more efficient. Chapter 5 summarizes what has been done during the thesis and presents the author's conclusions. The link to the GitHub repository and source code can be found in the appendix.

2 Background information

The data is gathered and processed which is then used to locate a person in the building using a Python script. This chapter explains some of the terminology and technology that is used to collect and save the surrounding information.

2.1 Python

Python is a high-level, interpreted and general-purpose programming language first created in the late 1980s. The first version was released in 1991, with the major releases 1.0, 2.0 and 3.0 released in 1994, 2000 and 2008 respectively. Each version improves upon the previous release by adding new features and fixing different flaws. Today, Python is the most popular programming language, with ~30% of all the programming language tutorial searches on google[1] and the second just behind Javascript as the language used for open source projects on GitHub[2]. It boasts a simple, beginner-friendly syntax, compared to most languages, and a large library of standards and toolkits, making it fairly easy to develop different applications without having to “reinvent the wheel”.

2.2 MySQL

MySQL is a Relational Database Management System (RDBMS). It was first released in 1995 and has been in continuous development since then. Based on miniSQL (mSQL), it was initially created for internal usage by David Axmark and Ulf Michael Widenius. In 1996 it was released to the public. MySQL quickly gained popularity as it kept its application programming interface (API) consistent with mSQL while also being faster and more flexible than its antecedent. This, along with being open source, as opposed to proprietarily licensed mSQL, gained the favour of many developers. [8]

MySQL is designed to be used for creating and managing relational databases. Like with other relational database management systems, it is possible to change, add or remove the data and its structures as well as answer queries. SQL statements (Structured Query Language) are used to communicate with MySQL databases. [8]

As of December 2020, MySQL is the second most used database management system right behind Oracle [9].

2.3 Wireless access point

Wireless access point (WAP) which is more often referred to as just access point (AP) is a physical networking device that allows other Wi-Fi capable devices to connect to a network. It connects directly to a wired network and provides wireless connections for other devices to

[1] https://pypl.github.io/PYPL.html

[2] https://madnight.github.io/githut/#/

use that wired connection. There are many different IEEE 802.11 standards for wireless access point and wireless router technology. They are constantly created to accommodate for the increasing need for faster connection [10]. Access points also have some limitations such as the amount of devices that can be connected to a single access point without the signal output degrading. Also the range of signal can vary greatly depending on the environment and should be taken into account when placing them across a certain area to provide service.

2.4 Network adapters

Network adapter (also known as network interface controller) is a computer hardware that connects a computer to a network. They can come in a form of built-in adapters (generally computer motherboards), expansion cards that connect to a motherboard, or external adapters that can be connected via USB-A to a computer. Depending on the type of the adapter, they can support either a wired connection, a wireless connection, or both. The external network adapters are usually used to receive a stronger signal from access points and discover them from further away and collect more data about a network than a regular network interface controller is capable of. [11] Because of that we will be also using an external network adapter to scan for access points. The used external adapter can be seen in Figure 1.

Figure 1. The external network adapter used for this thesis.

The specific one used is tp-link TL-WN822N ver 5 with RTL8192EU rev B chipset which supports monitor mode.

2.5 Beacon frame

Beacon frame is found among the management frames in wireless local area networks (WLAN) that are based on IEEE 802.11 standard. It is constantly emitting information about the access point, announcing its presence to nearby devices that they can connect to it. Beacon frames are usually transmitted in short intervals of 100 time units (1 TU = ~1ms) [12] and it contains all the information about the network. The most important ones for this thesis are the following:

- **Media access control address (MAC address)** is a unique address for each network and can be used to differentiate between networks even if their names are the same. However, it might not always be named as such, depending on the software and toolkit for scanning used. Most tools use the name BSSID (basic service set identifier) for this. [12]
- **Service set identifier (SSID)** is a network name that is usually in natural, human-readable language. Unlike network MAC addresses, this is usually customisable warranted that the person has administrative access to the network. [13]
- **Signal strength** indicates the strength of a network signal that is broadcasted by an access point. Usually, the higher the indicated signal, the closer is the access point and the less obstacles there are between the device and the access point. Obstacles can be walls, windows, doors, for example. This can considerably reduce the signal strength and should be taken into consideration when analysing signal strength. [14]

3 Program architecture and design

The program is developed in a way that it would be possible to execute it using either a command line interface (CLI) as well as a graphical user interface (GUI). As the program needs to connect to a database and use superuser rights for scanning, a settings file has also been created and needs to be filled beforehand if only using a CLI version. However, this part is optional when using the GUI as the fields there can be filled after the program has been executed. If the settings file is filled, these fields will be automatically filled in the latter case.

The graphical user interface is created using the Tkinter package for Python. Tkinter is a lightweight cross-platform GUI framework which can be used to set up GUI applications fairly easily [15]. However, a notable drawback for this package is that it can look quite outdated to some and therefore they may opt for another framework. The GUI consists of 3 different tabs.

The first one can be seen in Figure 2 and is used to define general settings.

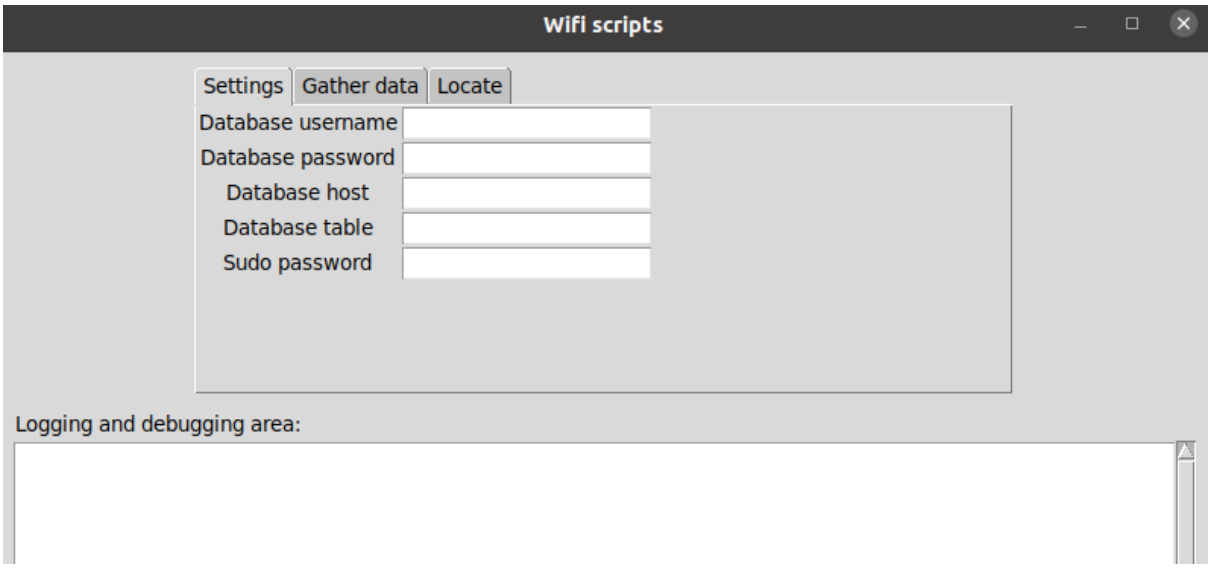

Figure 2. The settings tab.

In this tab it is possible to set the username and password which can be used to access the database. Along with these, the database host address and table can be specified where the data is retained. Finally, there is a textbox for setting the superuser password.

The second tab can be seen in Figure 3 and is for gathering data.

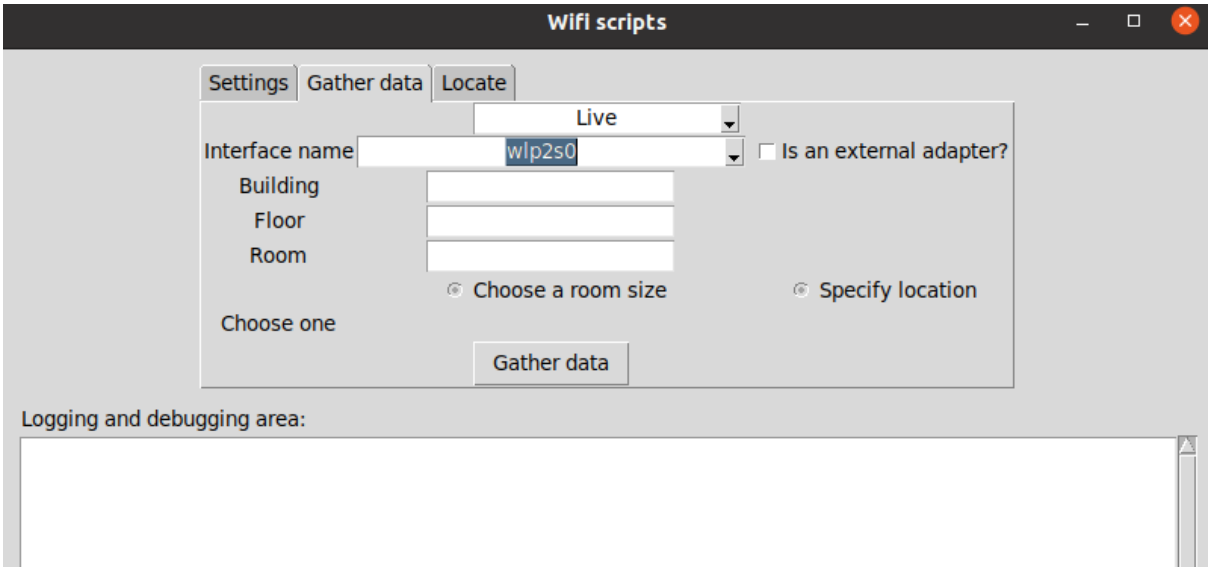

Figure 3. The data gathering tab.

It is possible to choose whether to use the live capture method or load data from a file. Depending on the choice, the user can then either choose the network interface to scan on or specify the name of the file. Building, floor and room name and room size or specific location have to be specified before the user is able to start gathering data.

The third tab is depicted in Figure 4 and is used to locate the person.

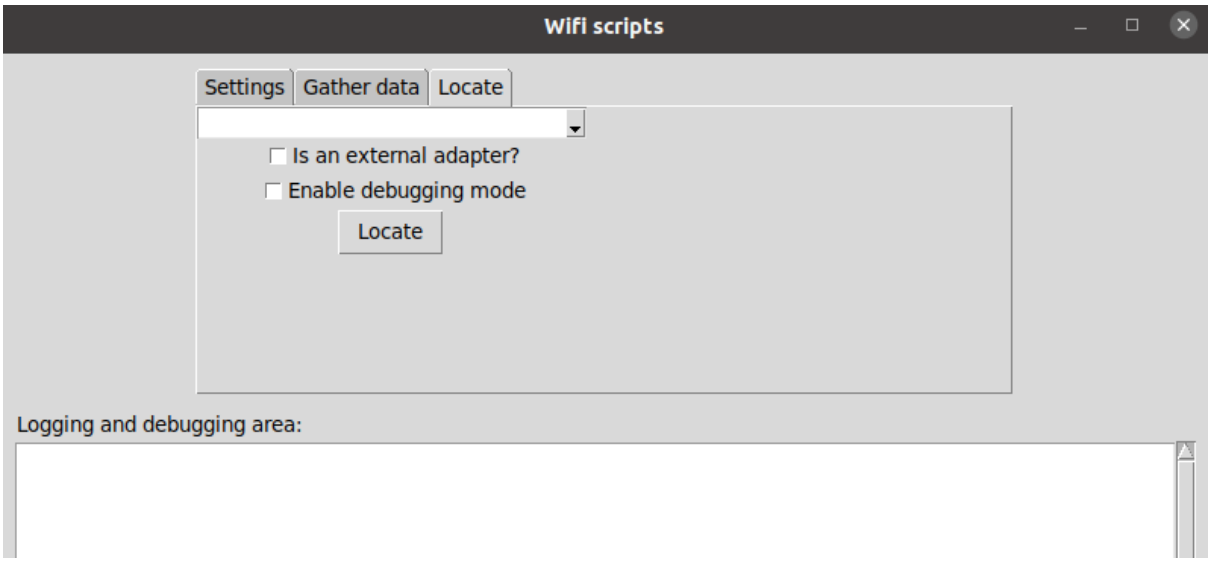

Figure 4. The locating tab.

The user needs to specify the interface and its type that is used for scanning and to compare against values in the database. A debugging mode has been incorporated into the tab so that

the user would have the possibility to see all the found access points and their signal strengths in the current location that are taken into account when the script makes the guess. This can be extremely helpful when trying to find out why the script returned a wrong answer and see the corresponding likelihood percentages of possible locations.

3.1 Database model

As the result of this thesis is a database with access points and their signal strengths located in the Delta building, a database model had to be developed to hold all the necessary data. The database consists of 5 different tables, which are:

- access_point. This table holds the information about the access points found in the building. The physical address (BSSID) and its current name at the time (SSID) is saved to the database for each access point. Although the SSID can change in time, the physical address does not and therefore is used to identify the access points when executing scans.
- place. Holds the data about the general area or room that is chosen by the user. The specified area, building and floor are saved in this table. It also holds the access point IDs which were found for each area and the signal strength ranges made up of the scans from multiple spots in the specified area. The data that is stored in this table can be used to locate a person in a room without giving an exact whereabouts of a person in that room.
- place_detail. This table holds the information about more specific locations in each area (e.g. center of the room, by the table, North-East corner etc.). It records the data about signal strengths at the given location for each access point and is then used to update the range for the given area in the place table. This table can be useful when trying to determine the user's exact position in conjunction with the place table if the room is known.
- place_without_adapter. This table is the same as the place table but holds the data about scans executed with the internal network adapter of the device.
- place_detail_without_adapter. This table is the same as the place_detail table but holds the data about scans executed with the internal network adapter of the device.

As seen in Figure 5, the fields in the database are classified as either INT or VARCHAR type. The INT type is used for numerical values, more specifically integers. For example, the building floor can only be an integer and therefore this field type is the most suitable for this. The place, place_detail and access_point IDs are also saved to the database as integers. Each time a new row is added to these tables, the ID of that row is determined by automatically incrementing 1 to the ID of the last added row. The VARCHAR type is used for fields that can contain multiple types of characters. A good example of this is a location field in these

tables, as its value can be a word (e.g. “corridor”) or a classroom number (e.g. “2030”) or a combination of both. The NOT NULL after the type indicates that the field value can not be empty when the row is added into the database.

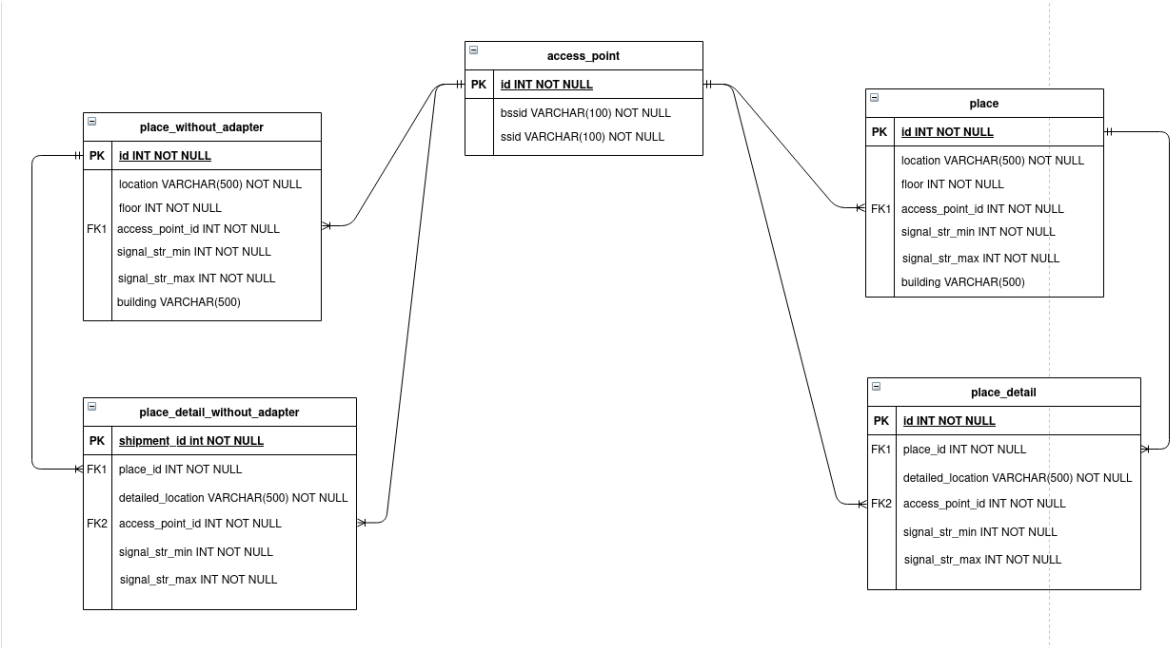

Figure 5. Database structure drawn using Diagrams[3].

An additional database function was created that is invoked when the user specifies where the data is being gathered at. The function determines the Levenshtein distance between the user’s input and existing names in the database. If the distance is less than or equal to 2 and higher than 0, the program suggests to the user possible names from the existing ones. The user then has the option to change it or keep the one they inserted as an input. If the location name matches exactly with one in the database, the user is notified that the location already exists in the database and the values for the given location will be updated as a result of the scan.

Levenshtein distance measures the difference between two sequences. The Levenshtein distance calculates the minimum number of single-character edits (insertions, deletions or substitutions) that are needed to change one word into the other. This distance was considered in 1965 by Vladimir Levenshtein who was a Soviet mathematician [16].

[3] https://app.diagrams.net/

```
function LevenshteinDistance(char s[1..m], char t[1..n])
  #d is a table with m+1 rows and n+1 columns
  declare int d[0..m, 0..n]
  #source prefixes can be transformed into empty string by
  #dropping all characters
  for i from 0 to m
      d[i, 0] := i

  #target prefixes can be reached from empty source prefix
  #by inserting every character
  for j from 0 to n
      d[0, j] := j

  for i from 1 to m
      for j from 1 to n
      {
          if s[i] = t[j] then cost := 0
                         else cost := 1
          d[i, j] := minimum(
                              d[i-1, j] + 1,     #deletion
                              d[i, j-1] + 1,     #insertion
                              d[i-1, j-1] + cost   #substitution
                            )
      }

  return d[m, n]
```

Figure 6. Pseudocode for Levenshtein distance [17].

The pseudocode for this algorithm is outlined in Figure 6.

3.2 Data gathering

To determine the current position of the script user, the building has to be mapped first. It is necessary to gather access point data from as many classrooms, hallways and student resting areas as possible to ensure that the locating script could give an accurate estimate of the user's whereabouts. The user can specify their building and floor, pick the room size, choose the name for the current spot and is given the instructions to map the room accordingly thereafter.

3.2.1 Methods

There are two different methods that can be used to capture and save data. The first option is to specify the file location and analyse the data from the file. The second option is to live capture beacon frames emitted by the access points. Both have their own advantages and disadvantages.

These methods utilise the pyshark[4] Python library which uses tshark for its functions. Tshark is a network protocol analyser that is used for capturing packet data from a live network and

[4] https://github.com/KimiNewt/pyshark/

reading them from previously saved files [18]. Therefore this library was chosen as it could do everything that was necessary for this project. Each beacon frame holds a myriad of information and is separated into multiple layers in a packet. Since layer names can overlap or in some cases not exist at all, each layer has to be checked individually if the necessary variables exist in that layer and to avoid any errors. More specifically, each layer is checked to find the access point's physical address, name and signal strength as these are the values we are interested in. An example of how data can be extracted from a beacon frame using pyshark can be viewed in Figure 7.

```
35      for packet in capture:
36          for layer in packet.layers:
37              if layer.get('signal_dbm'):
38                  signal_str = int(layer.get('signal_dbm'))
39              elif layer.get('bssid'):
40                  bssid = layer.get('bssid')
41              elif layer.get('wlan.ssid'):
42                  ssid = layer.get('wlan.ssid')
```

Figure 7. Code snippet for finding the necessary variables from layers.

That data is then used to compare against the created dictionaries with access points' physical addresses and signal strengths. If a physical address exists in the dictionary, the found minimum and maximum signal strengths are updated for that access point in the corresponding dictionaries, otherwise a new entry will be added to them. The collected data is then compared to the values existing in the database similarly to the dictionaries - If an access point is found in the database for the defined location, the minimum and maximum signal strengths are compared and updated accordingly, otherwise a new row is added to the database. The pseudocode for analysing data can be seen in Figure 8.

```
function analyseData(captured_packets):
  #declare empty dictionaries and variables
  ssid_dict, signal_str_min_dict, signal_str_max_dict = {}
  signal_str, bssid, ssid = None

  foreach packet in captured_packets:
      foreach layer in packet.layers:
          #all these variables are found in different layers
          if signal_strength found in layer:
              assign signal_strength to signal_str
          else if physical_address found in layer:
              assign physical_address to bssid
          else if access_point_name found in layer:
              assign access_point_name to ssid

      if key bssid does not exist in ssid_dict:
          add key bssid to ssid_dict with ssid as value

      if key bssid does not exist in signal_str_min_dict:
          add key bssid to signal_str_min_dict with signal_str as value
      else if signal_str is less than existing value in signal_str_min_dict: #compare the existing minimum values for the found bssid key
         update the signal_str_min_dict with signal_str value on bssid key

      if key bssid does not exist in signal_str_max_dict:
          add key bssid to signal_str_max_dict with signal_str as value
      else if signal_str is higher than existing value in signal_str_max_dict: #compare the existing maximum values for the found bssid key
         update the signal_str_max_dict with signal_str value on bssid key

      signal_str, bssid, ssid = None #reset variables for the next packet

    return ssid_dict, signal_str_min_dict, signal_str_max_dict
```

Figure 8. Pseudocode for analysing the packet data.

As tshark itself is only usable via the command line interface, it can be somewhat tedious to get the files' contents to just take a look at the data. A more user-friendly alternative for viewing the recorded data is using Wireshark[5] which is also a network analysing tool just like tshark. The difference being that Wireshark has a graphical user interface and that makes the inspection of contents in the file much easier. The contents of a packet is shown in Figure 9.

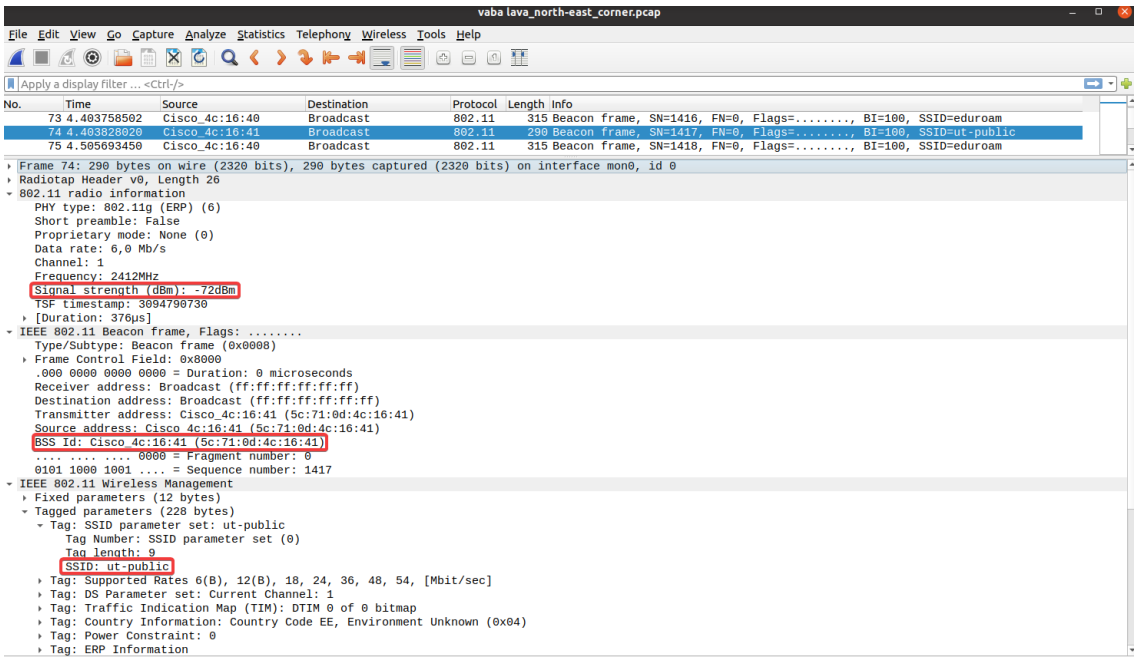

Figure 9. Viewing a single packet contents using Wireshark. Signal strength, BSSID and SSID can be seen in separate layers and are highlighted in the red.

[5] https://www.wireshark.org/

Using a previously saved file grants the user to choose a file or enter the name of a file that will be used to analyse and save the data. Currently there are set restrictions that allow only the files with .pcap or .pcapng extensions to be used as they are the most compatible with tshark and Wireshark. Compared to live capture, only one file, and therefore only one spot, can be analysed at once. This means that to insert the data from an area that has multiple scan files, each file has to be run manually each time. A Big advantage for this method, however, is that the user does not have to be on site to gather and compile the data if they have the required files. In addition to that, this method is a much faster way to insert data into the database as it skips the scanning part and jumps straight to reading the input.

If the live capture method is chosen for mapping a room, the user has a choice to either define a custom spot in the area or choose a room size from the following three options - small, medium or large. The user also has an option to define if they are using an external network adapter or an internal one.

When using the external network adapter for live capture, it is required to be set into monitor mode beforehand. Monitor mode allows the wireless network interface controller to monitor all the traffic transmitted by the access points without having to actually connect to them. This is necessary so that the interface could be able to capture the beacon frames. The script then determines how many packets to capture before analysing them. This is achieved by doing a pre-scan with the internal adapter and counting the amount of access points that are found. That amount is then multiplied by 50 to get the amount of packets that is to be captured before processing them. By multiplying the amount of access points by 50 ensures that the script can analyse about 50 packets sent out by each access point and therefore can provide more reliable data about signal strength ranges. The data gathering process takes about 5 seconds on average as the beacon frames are usually transmitted at the rate of 100ms / packet as mentioned previously. Therefore, a single access point may have up to about 200 packets analysed per room depending on the chosen room size and if the access point is visible to the external adapter in all the spots in the room. Once the scans are run and the packets are processed, the data is also saved to a file and is uploaded to a folder named *pcap_dumps* so that the data could be reused in the future.

When using the internal network adapter the script scans 10 times using the internal adapter. This is due to the script not collecting packets anymore as was done when using an external adapter. It uses the iwlist[6] tools for scanning which is part of the wireless tools set written for Linux kernel-based operating systems. However, running this command requires superuser rights to be able to find all the access points in the vicinity. Due to that the superuser password needs to be specified beforehand. As each scan takes about a second, the whole process of gathering data takes about 15 seconds when using an external adapter and about 10 seconds when using only an internal one.

[6] https://linux.die.net/man/8/iwlist

This method's advantage is that the data is not static and can be run multiple times if the user wishes to, updating the existing values in the database if needed, as opposed to reading data from a file. The disadvantage of this method can be that the data may not always be accurate as there can be multiple external factors that can interfere with the signal strengths of access points.

3.2.2 Scanning

The first option to use for scanning is defining a custom spot. In this case the script runs the scan only once in the chosen location and terminates after that. This can be a good option if the user wants to map an area that is not a room, but rather a hallway, for example. It could also be used to specify some more common spots in a larger room or auditory.

The second option is to choose a room size. There are no set instructions as to which size should be chosen at what time. The person is left to make that decision according to their own needs. I.e. the smaller spaces can also be mapped with large room parameters to get more thorough coverage of the room if the user wishes to. With each size, a different set of instructions are given to the user for mapping the room ranging from scanning once and up to four times.

When the small room size is chosen, the user is instructed to go to the center of the room and start the scan. After the scanning is done, the script terminates. This size is the most fitting for small office spaces and rooms up to $25m^2$.

When the medium size of the room is chosen, the user is instructed to go to opposite sides of the room and start scanning. This size could be used for smaller classrooms and rooms up to $50m^2$.

When the large size of the room is chosen, the user is instructed to go to each corner of the room and initiate a scan in each spot. The script terminates when the data from the last corner is saved to the database. This is advised to be used in auditories and rooms larger than $50m^2$, for example.

It is worth nothing that the users might not always have an external adapter at hand and have to use the built-in network adapter of their portable devices for locating. To help solve that problem, the scans are executed twice when using the external network adapter - the first time using an external adapter and the second time using the built-in network adapter. If the external adapter is absent when collecting data, only scans with the built-in one are initiated. This ensures that whichever option is chosen by the user the data will be saved to a corresponding database table.

3.3 Locating solution

Once the relevant data has been gathered it is possible to use it for locating a person in a building. A proof of concept script has been developed for the purpose of showing that the

collected data can be, in fact, used to locate a person inside the Delta building. The script gathers information about the surrounding access points and outputs the most probable location of the user based on that.

3.3.1 Gathering surrounding information

The surrounding information is gathered by scanning with either an external adapter or a built in one. This can be specified by the user before the script starts the scanning process. The scanning process is the same as described before in the gathering data section when doing a live capture but this time the scan is only initiated once instead of 10 times. When the scan is initiated, the command that is run returns data about surrounding access points. An example of the data received from an access point can be found in Figure 10.

```
Cell 04 - Address: 00:22:07:40:F7:07
          Channel:3
          Frequency:2.422 GHz (Channel 3)
          Quality=26/70  Signal level=-84 dBm
          Encryption key:on
          ESSID:"Liiva"
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                    24 Mb/s; 36 Mb/s; 54 Mb/s
          Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
          Mode:Master
          Extra:tsf=00002c66b970680b
          Extra: Last beacon: 712ms ago
          IE: Unknown: 00054C69697661
          IE: Unknown: 010882848B962430486C
          IE: Unknown: 030103
          IE: Unknown: 2A0104
          IE: Unknown: 32040C121860
          IE: IEEE 802.11i/WPA2 Version 1
              Group Cipher : TKIP
              Pairwise Ciphers (2) : CCMP TKIP
              Authentication Suites (1) : PSK
          IE: Unknown: 0B050300300000
          IE: Unknown: 42020000
          IE: Unknown: 2D1ABC081BFFFF000000000000000000000000000000000000
          IE: Unknown: 3D160308040000000000000000000000000000000000000000
          IE: Unknown: 4A0E14000A002C01C800140005001900
          IE: Unknown: 7F080500080000000040
          IE: Unknown: DD090010180203001C0000
          IE: WPA Version 1
              Group Cipher : TKIP
              Pairwise Ciphers (2) : CCMP TKIP
              Authentication Suites (1) : PSK
          IE: Unknown: DD180050F2020101840003A4000027A4000042435E0062322F00
```

Figure 10. Output about a single access point when running the *iwlist <interface> scan* command.

Figure 11 shows how the script then uses regular expressions to extract the access point's name, physical address and the received signal strength. Regular expression, or regex in short, is a specified search pattern in a sequence of characters that can be used to extract searched data [19].

```
cellNumberRe = re.compile(r"^Cell\s+(?P<cellnumber>.+)\s+-\s+Address:\s(?P<mac>.+)$")
regexps = [
    re.compile(r"^ESSID:\"(?P<essid>.*)\"$"),
    re.compile(r"^Protocol:(?P<protocol>.+)$"),
    re.compile(r"^Frequency:(?P<frequency>[\d.]+) (?P<frequency_units>.+) \(Channel (?P<channel>\d+)\)$"),
    re.compile(r"^Quality=(?P<signal_quality>\d+)/(?P<signal_total>\d+)\s+Signal level=(?P<signal_level_dBm>.+) d.+$"),
    re.compile(r"^Signal level=(?P<signal_quality>\d+)/(?P<signal_total>\d+).*$"),
]
```

Figure 11. Regular expressions to find the necessary data.

The script moves on to the decision making part about the user's location once the scanning is done.

3.3.2 Making the guess

Once the necessary information has been extracted by the script and saved into a list of values, it can then be used to compare to the values previously saved in the database. The database is queried for all the access points which were found during the scanning process, leaving out those not in vicinity. An example query can be seen in Figure 12.

```
SELECT p.location
FROM access_point ap
INNER JOIN place p ON ap.id=p.access_point.id
WHERE ap.bssid = bssid
AND signal_str BETWEEN p.signal_str_min AND p.signal_str_max
```

Figure 12. Query that is sent to the database for each found access point and its signal strength when using an external network adapter.

For each of the found access points, the signal strengths are compared. The row is then added to the list of possible results if it falls between the previously recorded minimum and maximum signal strengths. Finally the likelihoods for each location are calculated as percentages and the script outputs its prediction of where the user might be based on the highest occurrence of a possible location in the list. The pseudocode for finding the location of the user can be seen in Figure 13.

```
function findLocation():
  #declare empty dictionary
  possible_locations = {}

  scan once to get the data about the surrounding access points

  if no access points are found:
      return
  else:
      foreach access_point in scan_result:
          send a query to database with physical address and signal strength of an access point

          if query returns at least one result:
              #count the occurences
              foreach location in query_result:
                  if location exists in possible_locations:
                      #add one to existing value
                      possible_locations[location] += 1
                  else:
                      possible_locations[location] = 1

      if possible_locations is not empty:
          #return the location name which has the highest occurence
          return max(possible_locations)
      else:
          return
```

Figure 13. Pseudocode for determining the location of a user.

It is important to note that not all found access points may be listed in the database. This can be due to temporary hotspots created by other people or additional access points that may have been added between the timeframe where the data was gathered and used to locate the person.

4 Results

The scanning and gathering data about the Delta building took about 6 hours in total. 390 different access points were discovered in the process of scanning for networks and within that time frame 45 different rooms and areas were mapped. This includes most of the free access areas and classrooms in the building. The gathered data for one of the locations for two different access points can be seen in Figures 14 and 15. 318 out of the 390 discovered access points (81,5%) were either named “eduroam” or “ut-public” which are networks that all the students and staff can freely use and are more likely to be present for a long period of time. Most of the other networks that are not named, are set up specifically for use in certain classrooms and labs (i.e. Computer graphics lab and Robotics lab have separate networks set up). This indicates that the Delta building has very good network coverage and the Wi-Fi signal should be strong everywhere in the building for the publicly accessible networks. An interesting note is to be made about the efficiency of an external network adapter that was used when capturing access points, however. When looking at the data gathered in the database, the internal network adapter discovered almost 3 times more access points (3115 database entries) than the external adapter did (1224 database entries). It is possible that this difference derives from the used laptop’s built-in adapter being manufactured by a different company than the external one or the external adapter may have been manufactured earlier and may therefore not be as efficient. Another possibility is that the internal adapter might be able to see the access points that operate on the 5GHz band while the used external adapter can only see the access points on the 2.4 GHz band. If it is the latter case, in the future, the frequency field should be included in the database to be able to distinguish between the access points better.

	id	location	floor	access_point_id	signal_str_min	signal_str_max	building
Edit Copy Delete	397	2048	2	71	-85	-67	Delta
Edit Copy Delete	398	2048	2	72	-86	-67	Delta

Figure 14. Two of the access points found in the place table for room 2048.

	id	place_id ▲ 1	detailed_location	access_point_id	signal_str_min	signal_str_max
Edit Copy Delete	634	397	north-east corner	71	-85	-73
Edit Copy Delete	665	397	north-west corner	71	-83	-67
Edit Copy Delete	694	397	south-west corner	71	-85	-76
Edit Copy Delete	720	397	south-east corner	71	-85	-71
Edit Copy Delete	635	398	north-east corner	72	-84	-73
Edit Copy Delete	666	398	north-west corner	72	-83	-67
Edit Copy Delete	695	398	south-west corner	72	-85	-76
Edit Copy Delete	721	398	south-east corner	72	-86	-72

Figure 15. More specific locations and signal strengths for the two access points in room 2048.

Due to the pandemic there were not many people in the building which made the data capturing a fair bit easier in public areas. However, due to this, most of the lecturers and professors were not present either and therefore their personal offices have not been mapped. This is something that certainly needs to be done in the future to achieve full coverage of the building and to help pinpoint more accurately the location of the user.

The locating script was tested in the Delta building about two months after the data was recorded. This gave a good indication of how the previously collected information about the access points fares after some time has passed. The locating script was executed a total of 64 times while walking around the areas and classrooms in the Delta building. An example of the script's output is shown in Figure 16. The estimated location of the user was either correctly guessed, or guessed the user to be in the neighbouring room or area, 42 times (~66%). This means, 22 times the error was larger than that. When the locating script was used with an internal adapter, the script tended to be more accurate compared to when an external one was used. This is most likely due to the database having almost three times the entries for an internal adapter than there are for an external adapter. Therefore, when using the built-in adapter, the database returned more results which could be used to more accurately guess the location.

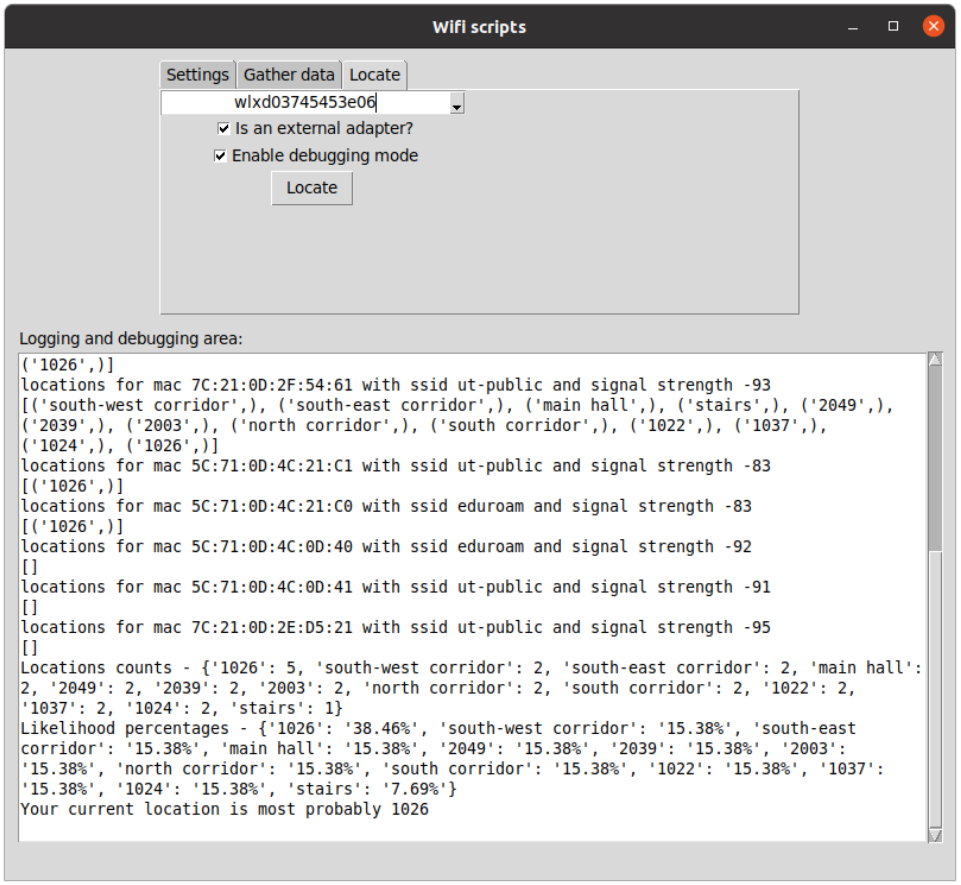

Figure 16. Example of the script's output when determining the user's location.

There were a few problems that were identified in the process of testing the locating script:

- In some cases, due to some area names overlapping on different floors, it was not possible to identify immediately if the user was put on the correct floor. This is something that could be fixed in the future with relative ease by also adding the floor number to the output alongside the location.
- Due to how the locating algorithm currently works, there were cases where the correct answer has the same amount of occurrences as an incorrect one. This created the problem that if the incorrect result was added into the possible locations before the correct one, the incorrect answer was picked. This is due to the *max()* function selecting the first occurrence of the maximum value in the data structure. There is no easy way to quickly fix that error and most likely needs a more thorough solution that does not use the *max()* function to choose the most likely place candidate.
- As the gathering of data and using it to locate was done within about two months apart, there were cases where an access point was found but it did not exist in the database or the signal strength was not between the recorded maximum and minimum values (see Figure 17). Therefore those could not be used to locate the user.

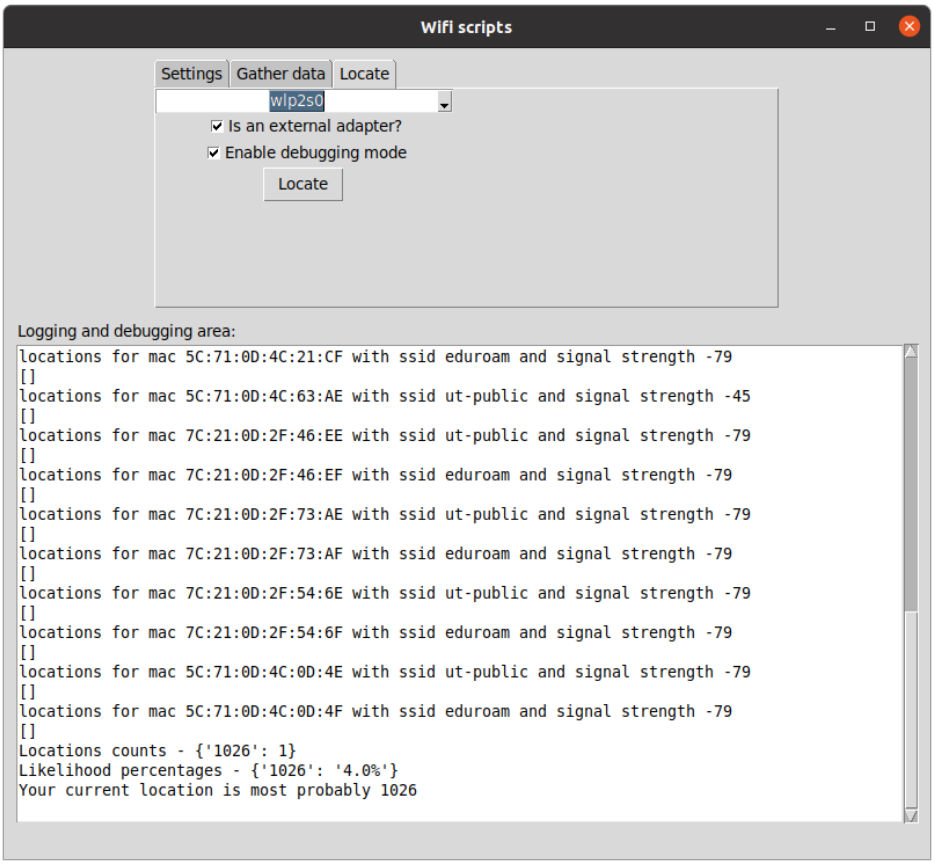

Figure 17. Case where almost no matching results were found in the database for the discovered access points.

Overall, given that the locating script was developed more as a proof of concept and did not use a very complex algorithm, the result in the end was surprisingly good. By using a more thought out algorithm for locating, it should be possible to achieve the accuracy of at least 80%. This can definitely be improved even further when the building is completely mapped and ideally achieves ~95% accuracy which could allow for the user's location to be estimated even more precisely than just in a room or an area level.

4.1 Problems and Limitations

Operating system - Currently the scripts are built to mostly work on Linux operating systems. That is due to the scripts using iwlist tools and one of the essential Python libraries for this project, pyshark, does not support Windows platform for live capture[7]. Although the script for gathering data could be run on a Windows operating system, the functionality is restricted to only doing it via file capture and could not be used for locating on the same system.

Missing rooms in database - Due to COVID-19 and auditory classes not happening during the period of the writing of the thesis, the access to classrooms, staff rooms and offices was heavily restricted. Although most of the classrooms and hallways have been mapped for the Delta building and are present in the database, most of the professors' and teaching assistants' cabinets were inaccessible as their offices could not be accessed via key card and very few of them were present at the time the data was recorded.

Adapter differences - As there are many different kinds of internal and external adapters, some differences in signal strengths and discoverable access points can vary to some extent. For example, in some cases internal adapters can pick up even more access points than the external ones. It can be dependent on the manufacturer, model and when the adapter was manufactured.

Interface not in monitor mode - One of the known problems for the gathering script is that it can become stuck when the chosen interface is not set into monitor mode before starting the live capture. This is due to script scanning for a set amount of beacon frames which can only be captured when the interface is in monitor mode. The script moves on to another block of code once the set amount has been captured. When the interface is not in monitor mode, the beacon frames can not be captured and the amount stays on 0 which results in the script running infinitely. The live capture function does have a timeout parameter which can be specified but due to that having a bug of crashing the program once the timeout has been reached even when using an interface that is in monitor mode, the author has opted to not use it in this case.

[7] https://github.com/KimiNewt/pyshark/blob/master/src/pyshark/capture/live_capture.py#L50

4.2 Future possibilities

Multiple operating systems support - As most of the functionality is currently restricted to Linux operating systems, one of the options could be to expand the functionality to Windows systems. This, however, could be quite a large undertaking as the pyshark library needs to be replaced with either something else that is compatible with Windows or to develop custom tools for this purpose. As for replacing iwlist tools, netsh could be used for Windows operating systems.

Interface recognition and monitor mode - One of the possibilities that could be improved on is the script automatically detecting the user's interface that is set to monitor mode or checking if the interface is in monitor mode before starting live capture to avoid the script becoming stuck. Another option would be adding a Bash script to set the chosen interface to use monitor mode, saving the hassle of having to manually do that beforehand.

Visual representation - When using the locating script, it outputs the current estimated location of the user. This can be somewhat confusing to some people as just the room number or hallway name may not say much to them. This could be improved by creating a visual output of where the user is located which could help them find their way around the Delta building easier.

Directions - One of the additions to current locating script and offered future possibilities is developing a way to get directions by the script on how they could reach their destination, given their starting point. This could be done by either using text or visually marking the route on the layout of the building.

Mobile app - As carrying around a heavy laptop and setting up the correct libraries to locate oneself can be extremely tedious, one could make this script and/or easier to use than it currently is. One option to make this more accessible to casual users, a mobile app could be created that uses the database to locate them. This way the user could just install the app on their phone and start using it straight out of the box and without any hassle.

Delta building fully mapped - Due to the access of the Delta building being somewhat restricted, one of the main goals should be to map the whole building in the future so that the estimated locations could be more precise for the users.

Automate setup process - Downloading all the correct packages and dependencies to get the scripts running can be tedious as this project uses multiple of them. The more packages there are, the more likely it is also to fetch the incorrect ones which could only cause more headaches. This could be improved upon by adding a Bash script or a Makefile that downloads the necessary packages and makes it easier for the user to start using these scripts. Another option is to create a Docker image which already has all the necessary dependencies and libraries. This way the user only has to have Docker installed and can instantly start using these scripts when they start the Docker image without any previous setup.

Improve the locating script algorithm - The locating algorithm used in this thesis is not very complex and therefore did not achieve a very high accuracy either. This is something that can definitely be improved on and maybe even incorporate machine learning in the future when trying to guess the location of the user.

5 Summary and Conclusion

In this thesis, a database with access points in the Delta building was created. To achieve that, a script for data gathering was developed in the process. The quality of the recorded data was later tested with a proof of concept locating script.

The data capturing was done by utilising the pyshark Python library which allowed the beacon frames to be captured and analysed. The user could specify different parameters to determine how the script would record data and the user was given corresponding instructions on where to go and what to do based on that. Almost 400 different access points were present in the Delta building at the time the data was captured. More than 80% of those were publicly accessible and could be used by the students and the staff. This means that those are the access points that are more likely to be static and can be relied on more for locating a person in the long term.

The locating script was tested against the available data a few months after it was gathered into the database. The results were surprisingly decent by managing to guess the user's whereabouts somewhat correctly $^{2}/_{3}$ of the time. By achieving greater coverage of the building, the accuracy of the locating script could possibly improve by quite a bit. A few problems became transparent during the testing as well. For example, the output could specify the floor of the user when making the guess to decrease the confusion whether the guessed location is correct when the location name exists on multiple floors.

In conclusion, although there were a few areas discovered that could be improved on in the process of developing and testing these scripts, the results were overall satisfactory and can be used in the future as a solid base for future development and ideas for improving the current work further.

6 References

[1] 9 Oldest Maps in the World: oldest.org. www.oldest.org/geography/maps/ (04.05.2021)

[2] Gatty H. Finding Your Ways Without Map or Compass. New York: Dover Publication. 1958.

[3] Guarnieri M. Once Upon a Time?The Compass [Historical]. *IEEE Industrial Electronics Magazine*, 2014, No. 2, pp 60-63.

[4] Oxley A. Chapter 2 - Introduction to GPS. *Uncertainties in GPS Positioning*. Academic Press, 2017, 19–38.

[5] Srivastava I. How Kargil Spurred India to Design Own GPS: The Times of India, 2014. https://timesofindia.indiatimes.com/home/science/how-kargil-spurred-india-to-design-own-gps/articleshow/33254691.cms (04.05.2021)

[6] After 13 Years, Galileo Satellite Navigation Complete at Last: De Ingenieur, 2018. www.deingenieur.nl/artikel/after-13-years-galileo-satellite-navigation-complete-at-last (04.05.2021)

[7] Chen Y, Kobayash H. Signal Strength Based Indoor Geolocation. *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, 2002, pp 436–439.

[8] Pachev S. Understanding MySQL Internals. O'Reilly Media. 2007.

[9] Liu S. Ranking of the Most Popular Database Management Systems Worldwide, as of December 2020*: Statista, 2020. www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/ (04.05.2021)

[10] What Are IEEE 802.11 Standards? : 802.11a/b/g/n/Ac/Ax: SignalBoosters, 2020. www.signalboosters.com/blog/ieee-802.11-standards-explained-802.11abgnacax/. (04.05.2021)

[11] Wiesen G. What is an External Network Adapter: EasyTechJunkie, 2021. https://www.easytechjunkie.com/what-is-an-external-network-adapter.htm (07.05.2021)

[12] Gast MS. 802.11 Wireless Networks: The Definitive Guide. 2nd ed. O'Reilly Media. 2005.

[13] Hoffman C. What Is an SSID, or Service Set Identifier?: How-To Geek, 2017. https://www.howtogeek.com/334935/what-is-an-ssid-or-service-set-identifier/ (07.05.2021)

[14] Tumusok JP, Newth JD. Wi-Fi Signal Strength: What Is a Good Signal And How Do You Measure It: eyeSaaS, 2018. https://eyesaas.com/wi-fi-signal-strength/ (07.05.2021)

[15] Tkinter library. https://docs.python.org/3/library/tkinter.html (07.05.2021)

[16] Levenshtein VI. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 1966, No. 8, pp 707–710.

[17] tommy. Levenshtein Pseudocode: GitHub, 2008. gist.github.com/tommy/14631 (04.05.2021)

[18] Tshark manual page. https://www.wireshark.org/docs/man-pages/tshark.html (07.05.2021)

[19] Homepage of Regular-Expressions. https://www.regular-expressions.info/ (07.05.2021)

7 Appendix

I Source code and Additional files

The source code for the scripts along with the instructions to run them are available on GitHub in this repository https://github.com/geioi/wifi-positioning

The included directory *delta_data* includes all the .pcap files that were generated when the data was gathered about the access points in the Delta building.

The included file *wifi_positioning_delta.sql* holds the processed data in database tables that can be used for locating.

II License

Non-exclusive licence to reproduce thesis and make thesis public

I, Geio Illus,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,
   Wi-Fi Positioning System
   supervised by Danielle Morgan.
2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Geio Illus
07/05/2021