Indoor Navigation

by

Lennart (XXXXXXX)
Jan (XXXXXX)
Eridy (XXXXXXX)
Andreas (XXXXXXX)

A project documentation submitted to

Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Service-centric Networking

Project Documentation

February 9, 2016

Supervised by:
Sebastian Zickau and Mathias Slawik

Service-centric NETWORKING

Technische Universität Berlin

Abstract

In this thesis, we show that lorem ipsum dolor sit amet.

Contents

1 Introduction

Introduction what is the goal of the project.

2 Related Work

The state of the art goes here. This includes:

- an overview over available technology
- a dicussion which technology is best to use
- CISCO MSE API wrapper test results

2.1 CISCO MSE API wrapper tests

In order to determine whether the CISCO MSE API wrapper provided by tubIT would be sufficient for the project's requirements we were asked to perform tests on it. Especially it was asked for details on how the API worked where, so what values could be retrieved via the API wrapper in which locations on campus and how precise the values would turn out to be.

Concerning use cases our project was focused on the mensa and the library, therefore we initially planned to be conducting tests in only these two locations. As the provided wrapper around the CISCO MSE API we had access to delivered back one short, simple XML line we decided to invest some time into developing a small tool which routinely queried the API for it's current status and saved the result into an easily readable JSON file for later investigation. The source code of the developed tool you can find in appendix 1.

It was planned to be conducting the tests on one day in the mensa and on another day in the library. On the first day we started around noon and ran the test tool on one of our notebooks connected to university WiFi, "eduroam". We started in the south-western corner near the windows, walked towards the south-eastern corner, went to the stairs in the northern part and upstairs and again at the window front to the south-western corner on the first floor. As it turned out, the results we got back were definitely not what we had hoped for, most importantly because longitude and latitude of the requesting user were missing completely. The following listing shows the first ten results logged in two second periods from the API wrapper:

```
1  {
2      "signal": [
3          {"timestamp": 1445344391, "latitude": "0.000000000000000", "
              longitude": "0.000000000000000", "building": "Mensa", "floor": "
              Mensa 1. OG"},
4          {"timestamp": 1445344393, "latitude": "0.000000000000000", "
              longitude": "0.000000000000000", "building": "Mensa", "floor": "
              Mensa 1. OG"},
```

```
5      {"timestamp": 1445344395, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "Mensa", "floor": "
           Mensa 1. OG"},
6      {"timestamp": 1445344397, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "Mensa", "floor": "
           Mensa 1. OG"},
7      {"timestamp": 1445344399, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "Mensa", "floor": "
           Mensa 1. OG"},
8      {"timestamp": 1445344401, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "Mensa", "floor": "
           Mensa 1. OG"},
9      {"timestamp": 1445344404, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "Mensa", "floor": "
           Mensa 1. OG"},
10     {"timestamp": 1445344406, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "Mensa", "floor": "
           Mensa 1. OG"},
11     {"timestamp": 1445344408, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "Mensa", "floor": "
           Mensa 1. OG"},
12     {"timestamp": 1445344410, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "Mensa", "floor": "
           Mensa 1. OG"},
13     ...
14   ]
15 }
```

Clearly it can be observed that the longitude and latitude values are unusable. Another take away was that the floor change during our test did not reflect into our captured results. Therefore we decided to directly test the library for comparable results. Inside the library, we started on ground floor and went upstairs in "circles" through the different levels. From the fourth floor we went back down straight forward. During that second test ten of the first fifteen responses from the API looked like:

```
1 {
2    "signal": [
3      {"timestamp": 1445345843, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "BIB", "floor": "
           Erdgeschoss"},
4      {"timestamp": 1445345845, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "BIB", "floor": "
           Erdgeschoss"},
5      {"timestamp": 1445345848, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "BIB", "floor": "
           Erdgeschoss"},
6      {"timestamp": 1445345850, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "BIB", "floor": "
           Erdgeschoss"},
7      {"timestamp": 1445345852, "latitude": "0.000000000000000", "
           longitude": "0.000000000000000", "building": "BIB", "floor": "
           Erdgeschoss"},
```

```
 8          {"timestamp": 1445345854, "latitude": "0.000000000000000", "
               longitude": "0.000000000000000", "building": "BIB", "floor": "
               Erdgeschoss"},
 9          ...
10          {"timestamp": 1445345870, "latitude": "0.000000000000000", "
               longitude": "0.000000000000000", "building": "BIB", "floor": "1.
                Obergeschoss"},
11          {"timestamp": 1445345872, "latitude": "0.000000000000000", "
               longitude": "0.000000000000000", "building": "BIB", "floor": "1.
                Obergeschoss"},
12          {"timestamp": 1445345874, "latitude": "0.000000000000000", "
               longitude": "0.000000000000000", "building": "BIB", "floor": "1.
                Obergeschoss"},
13          {"timestamp": 1445345876, "latitude": "0.000000000000000", "
               longitude": "0.000000000000000", "building": "BIB", "floor": "1.
                Obergeschoss"},
14          ...
15      ]
16  }
```

First, longitude and latitude were again unusable. This time though the floor information worked quite reliably and indicated very fast on which floor we currently measured. After that we were wondering whether eventually we would get back longitude and latitude values anywhere on campus and decided to give it one last try by taking one more measurement in the MAR building (Marchstraße).

One more measurement turned into three as during the first two attempts we got sudden disconnects and therefore unusable results. We walked the whole foyer from north to south side and somewhere near the entrance we suspect the wireless signal got bad and our notebook conducting the tests disconnected. In the third try though we finally were able to get back usable results, in which chosen ten results logged looked like this:

```
 1  {
 2      "signal": [
 3          {"timestamp": 1445350550, "latitude": "52.516903688639005", "
               longitude": "13.323958376544699", "building": "MAR", "floor": "
               Erdgeschoss"},
 4          {"timestamp": 1445350552, "latitude": "52.516903688639005", "
               longitude": "13.323958376544699", "building": "MAR", "floor": "
               Erdgeschoss"},
 5          {"timestamp": 1445350554, "latitude": "52.516903688639005", "
               longitude": "13.323958376544699", "building": "MAR", "floor": "
               Erdgeschoss"},
 6          {"timestamp": 1445350557, "latitude": "52.516903688639005", "
               longitude": "13.323958376544699", "building": "MAR", "floor": "
               Erdgeschoss"},
 7          ...
 8          {"timestamp": 1445350686, "latitude": "52.516864921942748", "
               longitude": "13.323953890659670", "building": "MAR", "floor": "
               Erdgeschoss"},
 9          {"timestamp": 1445350688, "latitude": "52.516864921942748", "
               longitude": "13.323953890659670", "building": "MAR", "floor": "
```

```
         Erdgeschoss"},
10       {"timestamp": 1445350690, "latitude": "52.516864921942748", "
             longitude": "13.323953890659670", "building": "MAR", "floor": "
             Erdgeschoss"},
11       ...
12       {"timestamp": 1445350845, "latitude": "52.516402095317481", "
             longitude": "13.323531401046099", "building": "MAR", "floor": "
             Erdgeschoss"},
13       {"timestamp": 1445350847, "latitude": "52.516402095317481", "
             longitude": "13.323531401046099", "building": "MAR", "floor": "
             Erdgeschoss"},
14       {"timestamp": 1445350849, "latitude": "52.516402095317481", "
             longitude": "13.323531401046099", "building": "MAR", "floor": "
             Erdgeschoss"},
15       ...
16    ]
17 }
```

Finally we received some longitude and latitude values. Unfortunately the three different pairs of longitude and latitude above were the only ones we could observe during the whole walk from north end to south end of the foyer, thus still rather unusable values.

In the end, our conclusion at that point in the project progress was to use the CISCO MSE API wrapper provided by the tubIT in order to retrieve the rough position of a user. This means the building and floor the request originated from. We recommended back to our supervisors not to use this API for receiving longitude and latitude as these values were either quite imprecise or not available at all. [**ioslINavGitHub**]

3 Concept and Design

Describe the concept and the design of your work here

4 Implementation

Only a rought sum up of the implementation. We could show the API here.

5 Evaluation

We dont realy have a evaluation right?

6 Conclusion

We did backend, ios, android, docker, nodejs and some other random stuff and now it works

List of Tables

List of Figures

Appendices

1 Tool for performing tests on the CISCO MSE API wrapper

```go
package main

import (
    "fmt"
    "log"
    "os"
    "syscall"
    "time"

    "encoding/xml"
    "io/ioutil"
    "net/http"
    "os/signal"
)

type WiFiInfo struct {
    XMLName          xml.Name `xml:"Info"`
    ChangedOn        string   `xml:"changedOn,attr"`
    ConfidenceFactor float32  `xml:"confidenceFactor,attr"`
    Building         string   `xml:"building,attr"`
    Floor            string   `xml:"floor,attr"`
    Network          string   `xml:"WLAN-Status,attr"`
    UserName         string   `xml:"username,atrr"`
    Longitude        float64  `xml:"lon,attr"`
    Latitude         float64  `xml:"lat,attr"`
}

type LogWiFiStruct struct {
    Timestamp int     `json:"timestamp"`
    Latitude  float64 `json:"latitude"`
    Longitude float64 `json:"longitude"`
    Building  string  `json:"building"`
    Floor     string  `json:"floor"`
}

func handleUserExit(signalChannel chan os.Signal) {

    for _ = range signalChannel {

        // Define a useful result file name (format: "wifi-measurement-year-
            month-day-hour-minute-seconds.json")
        fileName := fmt.Sprintf("wifi-measurement-%s.json", fileNameTime)

        // Open measurement result file
        fileResult, fileError := os.OpenFile(fileName, os.O_CREATE|os.O_RDWR|os.
            O_APPEND, 0666)

        if fileError != nil {
            log.Fatal(fileError)
        }
```

```go
49
50        // Close open logging file
51        defer fileResult.Close()
52
53        log.Printf("\nWritten measurement values to file %s. Good bye.\n",
              fileName)
54
55        os.Exit(0)
56    }
57 }
58
59 func main() {
60
61     // Put in here the API endpoint to the wifi location service.
62     const apiURL = "PUT THE API ENDPOINT IN HERE"
63
64     // Get time data
65     startTime := time.Now()
66     fileNameTime := startTime.Format("2006-1-2-3-4-5")
67
68     // Define a clean up channel
69     signalChannel := make(chan os.Signal)
70     signal.Notify(signalChannel, os.Interrupt, syscall.SIGTERM)
71     go handleUserExit(signalChannel)
72
73     log.Printf("Starting to measure WiFi API.\n")
74
75     for {
76
77         // Make a GET call on that URL
78         apiResp, apiError := http.Get(apiURL)
79
80         if apiError != nil {
81             log.Fatal(apiError)
82         }
83
84         // Read in all body content we got and close connection
85         xmlData, ioError := ioutil.ReadAll(apiResp.Body)
86         apiResp.Body.Close()
87
88         if ioError != nil {
89             log.Fatal(ioError)
90         }
91
92         // Our go struct representation of the tubIT XML
93         wInfo := WiFiInfo{}
94
95         // Parse received XML into struct
96         xmlError := xml.Unmarshal([]byte(xmlData), &wInfo)
97
98         if xmlError != nil {
99             log.Fatal("XML parsing error: %v.\n", xmlError)
```

```go
100            }
101
102            // Get the current UNIX epoch timestamp for logging
103            timeResult := time.Now().Unix()
104
105            // Build up the log string
106            logResult := fmt.Sprintf("\t\t{\"timestamp\": %d, \"latitude\": \"%.15f
                  \", \"longitude\": \"%.15f\", \"building\": \"%s\", \"floor\": \"%s
                  \"},\n", timeResult, wInfo.Latitude, wInfo.Longitude, wInfo.Building,
                   wInfo.Floor)
107
108            // Write log string to opened file
109            fileResult.WriteString(logResult)
110
111            log.Printf("Received: \"long\": %v, \"lat\": %v.\n", wInfo.Longitude,
                  wInfo.Latitude)
112
113            // Let the execution wait for 2 seconds
114            time.Sleep(2 * time.Second)
115        }
116  }
```