Comunicações Móveis

FEUP | MIEEC / MIEIC

01 March 2018

USER-AWARE FLYING AP

PROJECT PROPOSAL

Eduardo Almeida

eduardo.n.almeida@inesctec.pt

Introduction

- User-aware Flying AP (FAP)
  - Provides Wi-Fi connectivity to the users
  - Dynamic and autonomous positioning, in order to minimize the average distance to all users
  - Users have an Android App instaled on their devices reporting their positions

FAP

USER

USER

USER

Objectives

Develop the User-Aware Flying AP (FAP)

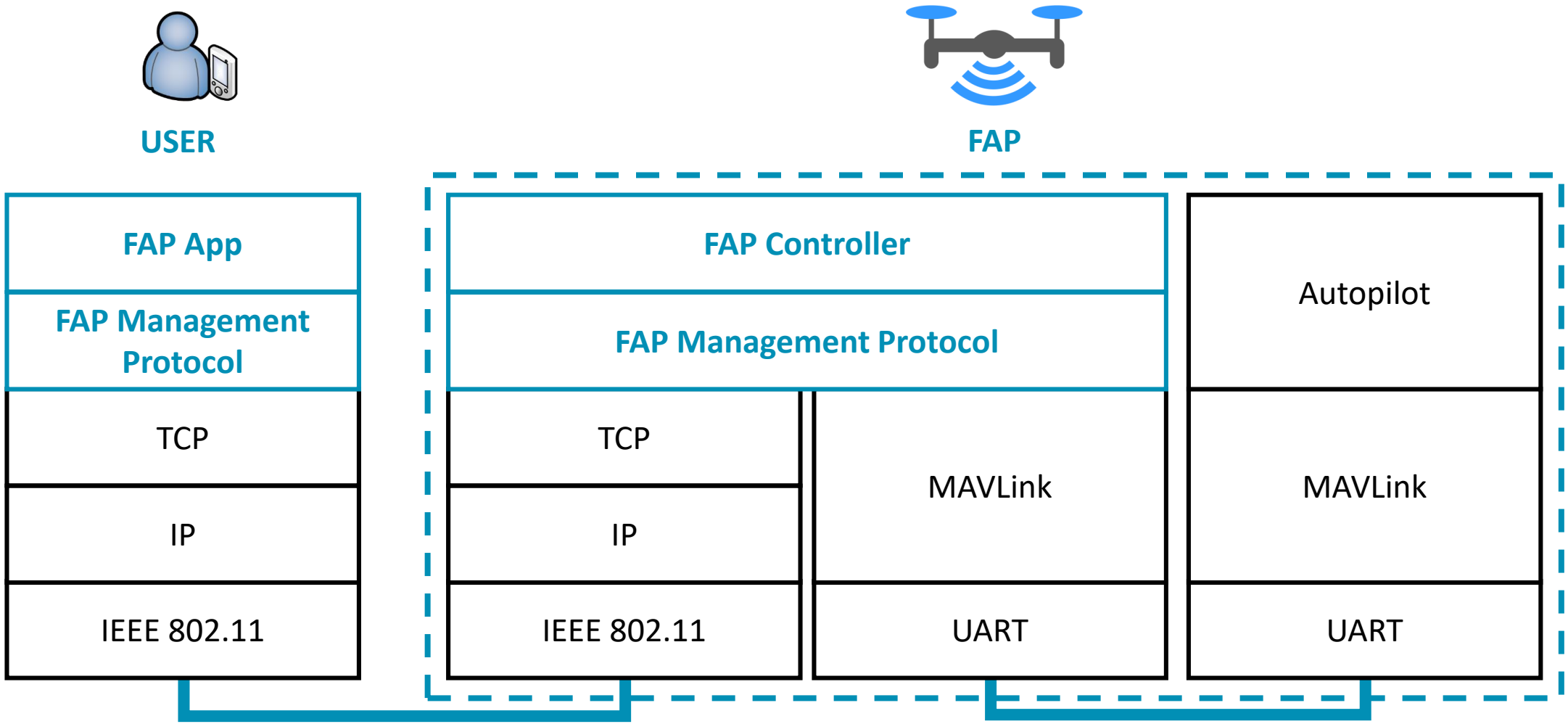
- Develop the FAP Controller
- Configure the AP
- Develop the Android App
- Develop the App – FAP Controller communications protocol

System Architecture | Overview

- Periodically sends GPS coordinates to FAP

- Determines position which minimizes the average distance to all users
- Moves to that position

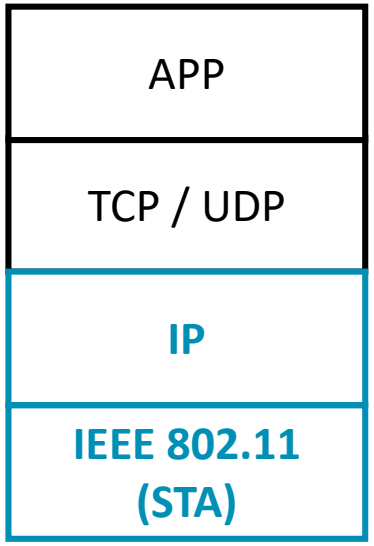
System Architecture | Control Plane

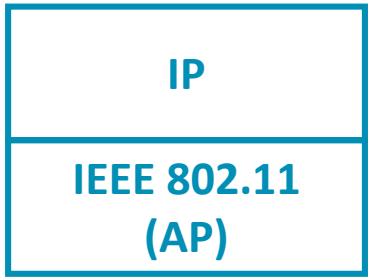


System Architecture | Data Plane

USER

FAP





Project Tasks

Task 1 FAP Android App

- Develop app's UI / UX
- Get GPS coordinates and periodically send them to FAP
- Automatically configure the Android device

Task 2 FAP Management Protocol

- Implement the FAP Management Protocol
- Manage user associations

Task 3 FAP Controller + AP

- Develop the FAP Controller
- Configure the AP

Task 1 | FAP Android App

- Develop the app's UI / UX
- Get GPS coordinates from the device and periodically send them to the FAP
- Automatically configure the Android device
  - Enable Wi-Fi
  - Enable GPS
  - Automatically associate to the FAP's SSID (Data Plane)
  - Request association to the FAP Controller (Control Plane)

Task 2 | FAP Management Protocol

- Implement the FAP Management Protocol
  - Client – Server protocol
  - Buffer received users' GPS coordinates and provide them to the FAP Controller
  - Validate GPS coordinates received from the users
  - Send MAVLink message to move the FAP to target coordinates (at the FAP Controller's request)
  - Developed in Java (Client) and C (Server)

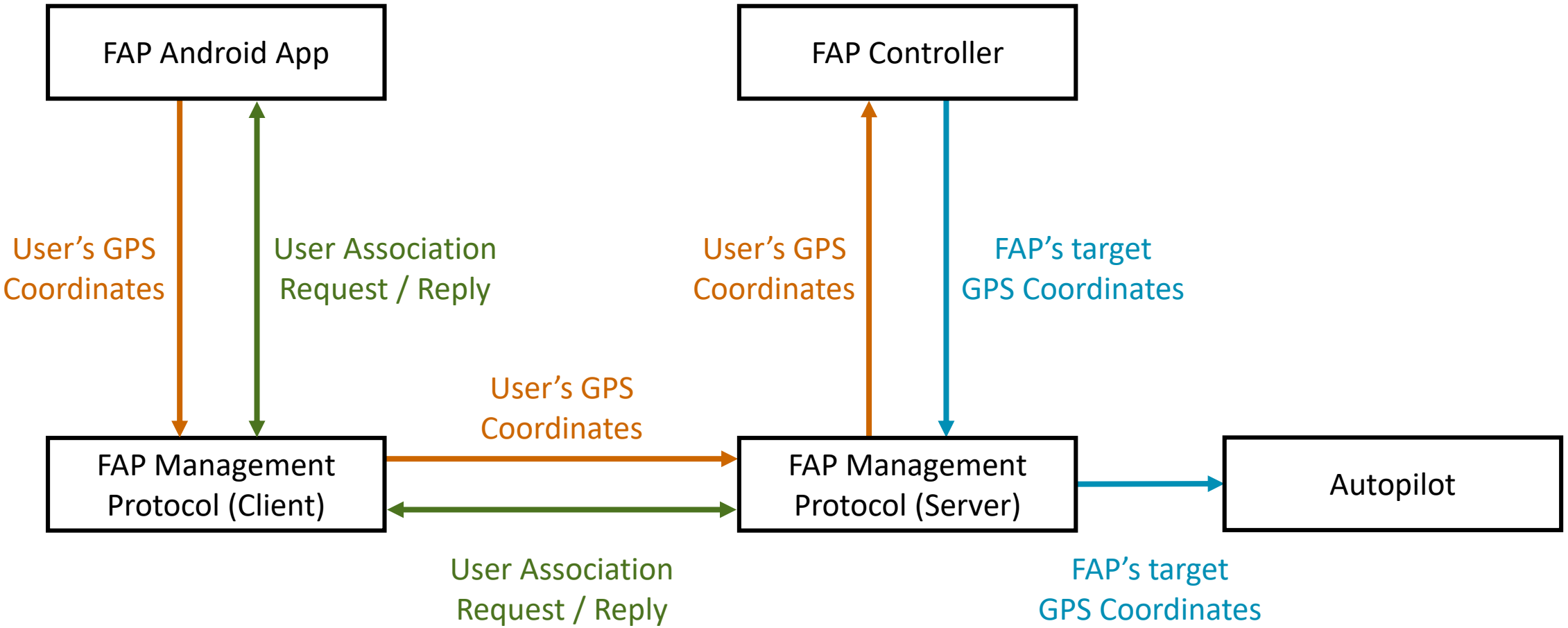
- Manage user associations
  - Keep track of the associated users
  - Accept / reject user association requests
    - The number of associated users should be limited (so that QoS is maintained)

Task 3 | FAP Controller + AP

- Develop the FAP Controller
  - Get the GPS coordinates of all users from the FAP Management Protocol
  - Determine the FAP's updated coordinates, as the coordinates which minimize the average distance to all users
  - Request the FAP Management Protocol to move the FAP to target coordinates
  - Developed in C and implemented on a Raspberry Pi 2

- Configure the AP
  - Linux OS
  - Configure Linux networking services (hostapd, DHCP Server, Network Address Translation)
  - Routing to backhaul network

System Interfaces



Student Groups

- 3 groups of 2 students
- Each group will develop one of the project's tasks
- Project's due date
  - Mid-April

- If you are interested in developing this project, please send an email
  - Eduardo Almeida [eduardo.n.almeida@inesctec.pt]
  - Indicate your preferred task(s) in priority
  - Until 07 March 2018

PROJECT GUIDELINES

Important Dates

- Project start
  - 19/March
- Submission of the preliminary code
  - By email to Eduardo Almeida [eduardo.n.almeida@inesctec.pt] in .zip format until 11/April.
- Submission of the final code
  - By email to Eduardo Almeida [eduardo.n.almeida@inesctec.pt] in .zip format until ~~25/April~~ 29/April (extended). An updated version of the code may be submitted by email until 16/May.
- Final demonstration of the project
  - 26/April or 30/April (depending on your "turma")
- Short report
  - To be submitted via Moodle one week after the Final Demonstration (~~03/May or 07/May~~) 14/May (extended)

FAP Android App | Requirements

- (1) Display a button (“Associate to FAP”) to associate the device to the FAP
- Upon pressing the button “Associate to FAP”
  - The app should automatically configure the device without user interaction (but requesting user permission)
  - Request the FAP Management Protocol to associate to the FAP
    - If accepted, start sending its GPS coordinates
    - If rejected, disassociate from the FAP, cancel configurations performed by the App, and goto (1)
- Periodically send GPS coordinates ($T_{GPS\_COORDINATES\_UPDATE}$ = 10 s)
  - If ACK is not received, disassociate from FAP, cancel configurations performed by the App, and goto (1)
- Display a button (“Desassociate from FAP”) to disassociate from the FAP
  - Request the FAP Management Protocol to disassociate from the FAP
  - Disassociate from FAP, cancel configurations performed by the App, and goto (1)
- The app should be able to run in the background, so the user can browse the Internet using other apps installed on their device
- Development and test
  - Android Studio emulator
  - Real Android smartphone (for the first and final demonstrations)

FAP Android App | Submission Instructions

- .zip file with the source code
  - Note: Please use the Android Studio's "File > Export to Zip File" option

- By email to Eduardo Almeida [eduardo.n.almeida@inesctec.pt]

FAP Management Protocol | Requirements

- The device may only send GPS coordinates once the association request is accepted
- If the server does not receive a GPS coordinates update in $2*T_{GPS\_COORDINATES\_UPDATE}$, assume the device is desassociated
- If the client does not receive ACK from server, assume the device is desassociated
  - In both cases, the Android App will have to start the association process again
- The users' GPS coordinates must be validated on the server
  - Considering a maximum distance from the FAP of 300 m
- The server may only accept up to 10 users simultaneously

FAP Management Protocol | Codebase

- Client (Java)
  - Codebase
    - src/ — Source code files
    - test/ — Tests to the code
    - .editorconfig — Config file for supported code editors / IDEs
    - README.md — Readme file

FAP Management Protocol | Codebase

- Server (C)
  - Codebase
    - bin/ — Executable
    - lib/ — Libraries (i.e., MAVLink, MAVLink emulator and JSON parser)
    - src/ — Source code files
    - test/ — Tests to the code
    - .editorconfig — Config file for supported code editors / IDEs
    - Makefile — Makefile
    - README.md — Readme file
  - Instructions
    - Build project: `$ make`
    - Run tests: `$ make run_test`

FAP Management Protocol | Public API

- The FAP Management Protocol's public API provides the protocol's services to the corresponding upper modules (FAP Android App and FAP Controller)
- Every function returns either a `boolean` or an `int`
  - return true; | return 0; → OK
  - return false; | return -1; → ERROR
- FAP Management Protocol's server address
  - IP: 10.0.0.254
  - Port: 40123
  - Protocol: TCP

FAP Management Protocol | Public API

- Client (Java)
  - `public boolean requestUserAssociation();`
  - `public boolean requestUserDesassociation();`
  - `public boolean sendGpsCoordinatesToFap(GpsCoordinates gpsCoordinates);`

N.B.: Refer to the API's documentation in the provided codebase.

FAP Management Protocol | Public API

- Server (C)
  - `int initializeFapManagementProtocol();`
  - `int terminateFapManagementProtocol();`
  - `int moveFapToGpsNedCoordinates(const GpsNedCoordinates *gpsNedCoordinates);`
  - `int getFapGpsNedCoordinates(GpsNedCoordinates *gpsNedCoordinates);`
  - `int getAllUsersGpsNedCoordinates(GpsNedCoordinates *gpsNedCoordinates, int *n);`

N.B.: Refer to the API's documentation in the provided codebase.

FAP Management Protocol | Protocol

- JSON format [1, 2]
  - A JSON parser is already included in the lib/ folder for C [3]. Yet, you are free to use other JSON libraries ([2] contains a list of libraries in C), but you will need the make the appropriate modifications.
  - Java has native libraries.
- Mandatory parameters in every message
  - “userId”: Last octet of the device’s IPv4 address (from 1 to 254)
  - “msgType”: Type of the message, whose values may only be
    - 1: USER_ASSOCIATION_REQUEST
    - 2: USER_ASSOCIATION_ACCEPTED
    - 3: USER_ASSOCIATION_REJECTED
    - 4: USER_DESASSOCIATION_REQUEST
    - 5: USER_DESASSOCIATION_ACK
    - 6: GPS_COORDINATES_UPDATE
    - 7: GPS_COORDINATES_ACK

[1] https://www.w3schools.com/js/js_json_syntax.asp
[2] http://www.json.org/
[3] https://github.com/udp/json-parser and https://github.com/udp/json-builder

FAP Management Protocol | Messages

User Association Request (Client → Server)

```
{
  “userId”: <USER_ID [INT]>,
  “msgType”: USER_ASSOCIATION_REQUEST
}
```

User Association Reply (Server → Client)

```
{
  “userId”: <USER_ID [INT]>,
  “msgType”: USER_ASSOCIATION_ACCEPTED |
             USER_ASSOCIATION_REJECTED
}
```

N.B.: “msgType” values as defined in the protocol specification

FAP Management Protocol | Messages

User Desassociation Request (Client → Server)

```
{
  “userId”: <USER_ID [INT]>,
  “msgType”: USER_DESASSOCIATION_REQUEST
}
```

User Desassociation Ack (Server → Client)

```
{
  “userId”: <USER_ID [INT]>,
  “msgType”: USER_DESASSOCIATION_ACK
}
```

N.B.: “msgType” values as defined in the protocol specification

FAP Management Protocol | Messages

GPS Coordinates Update (Client → Server)

```
{
  “userId”: <USER_ID [INT]>,
  “msgType”: GPS_COORDINATES_UPDATE,
  “gpsCoordinates”: {
    “lat”: <LAT_IN_DEGREES [FLOAT]>,
    “lon”: <LON_IN_DEGREES [FLOAT]>,
    “alt”: <ALT_IN_METERS [FLOAT]>,
    “timestamp”: <TIMESTAMP_ISO8601
                  [STRING]>
  }
}
```

GPS Coordinates Ack (Server → Client)

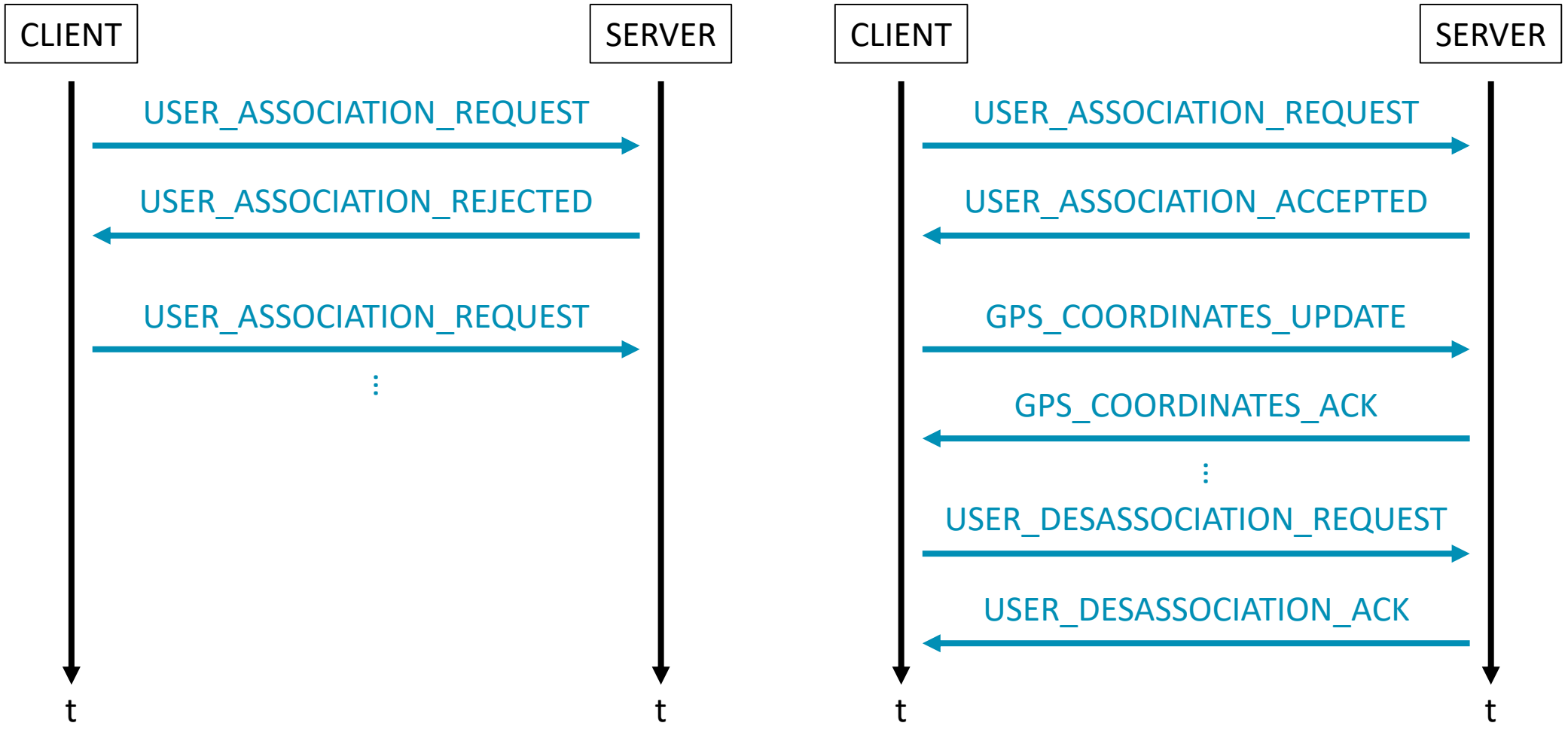
```
{
  “userId”: <USER_ID [INT]>,
  “msgType”: GPS_COORDINATES_ACK,
  “gpsTimestamp”: <TIMESTAMP_ISO8601
                   [STRING]>
}
```

N.B.: “gpsTimestamp” refers to the timestamp of the corresponding GPS_COORDINATES_UPDATE message

N.B.: Timestamp formated according to ISO 8601 (e.g., “2018-03-19T15:34:53Z”)

N.B.: “msgType” values as defined in the protocol specification

FAP Management Protocol | Messages



FAP Management Protocol | MAVLink

- The objective is to simply send the appropriate MAVLink messages to the Autopilot
- The Autopilot requires a HEARTBEAT message every 0.5 seconds
- North-East-Down (NED) format (x, y, z) vs. RAW GPS coordinates (lat, lon, alt)

- The official MAVLink C library is already included in lib/ folder
  - To simplify this task, the provided "MavlinkEmulator.h" library should be used instead

FAP Management Protocol | MAVLink

- MAVLink protocol
  - Standard – https://mavlink.io/en/
  - GitHub repository – https://github.com/mavlink/
    - C library MAVLink v2
      - https://github.com/mavlink/c_library_v2
      - https://mavlink.io/en/getting_started/use_source.html
      - https://github.com/mavlink/c_uart_interface_example

Message #	Name	Description (From Documentation)	URL
0	HEARTBEAT	"The heartbeat message shows that a system is present and responding."	https://mavlink.io/en/messages/common.html#HEARTBEAT
32	LOCAL_POSITION_NED	"The filtered local position (…) (aeronautical frame, NED convention)."	https://mavlink.io/en/messages/common.html#LOCAL_POSITION_NED
49	GPS_GLOBAL_ORIGIN	"(…) this message announces the origin (0,0,0) position."	https://mavlink.io/en/messages/common.html#GPS_GLOBAL_ORIGIN
84	SET_POSITION_TARGET_LOCAL_NED	"Sets a desired vehicle position in a local NED coordinate frame."	https://mavlink.io/en/messages/common.html#SET_POSITION_TARGET_LOCAL_NED

FAP Management Protocol | MAVLink Emulator

- `int initializeMavlink();`
- `int terminateMavlink();`
- `int sendMavlinkMsg_heartbeat();`
- `int sendMavlinkMsg_localPositionNed(GpsNedCoordinates *gpsNedCoordinates);`
- `int sendMavlinkMsg_gpsGlobalOrigin(GpsRawCoordinates *originRawCoordinates);`
- `int sendMavlinkMsg_setPositionTargetLocalNed(const GpsNedCoordinates *gpsNedCoordinates);`

- Every function returns an `int`
  - return 0; → OK
  - return -1; → ERROR

N.B.: Refer to the API's documentation in the provided codebase.

FAP Management Protocol | Submission Instructions

- Client (Java)
  - .zip file with the source code
- Server (C)
  - .zip file with the source code
  - Note: Before zipping, please delete the executable file from the codebase by running the following command
    - `$ make clean`

- By email to Eduardo Almeida [eduardo.n.almeida@inesctec.pt]

FAP Controller + AP | Requirements

- FAP Controller
  - FAP Controller should periodically determine the FAP's updated position ($T_{FAP\_POSITION\_UPDATE} = T_{GPS\_COORDINATES\_UPDATE} = 10$ s)
  - Get the GPS coordinates from all associated users from the FAP Management Protocol
  - If there are users associated
    - Determine the FAP horizontal position (x, y) which minimizes the average distance to all users (the FAP's altitude should be maintained).
    - Request the FAP Management Protocol to move to the target position
  - If there are no users associated
    - Maintain current position (i.e., do nothing)

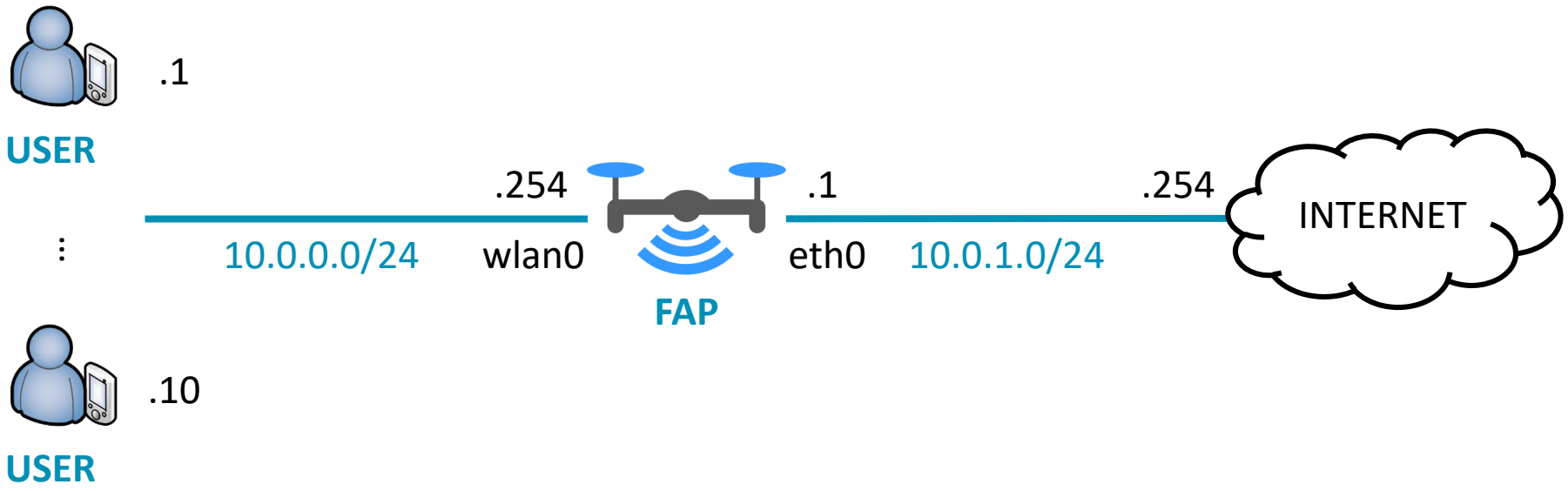
- Development and test
  - FAP Controller developed and tested on a Linux PC

FAP Controller + AP | Requirements

- AP
  - Configure Linux networking services: hostapd, DHCP server, NAT and default GW
    - SSID: “USER-AWARE-FAP”
    - Security: WPA2-PSK [Passphrase: “UserAwareFAP”]
    - FAP network address: 10.0.0.0/24
    - Default GW: 10.0.0.254
    - DNS server: 10.0.1.254
  - The corresponding services are already installed on the provided Raspberry Pi (hostapd, isc-dhcp-server, iptables, route)
  - Configurations must be persistent (i.e., the device must be operable without human interaction upon powering it up)

- Development and test
  - AP implemented and tested on a Raspberry Pi 2 with USB Wi-Fi dongle

FAP Controller + AP | Requirements



FAP Controller + AP | Codebase

- Codebase
  - bin/ — Executable
  - lib/ — Libraries (i.e., FAP Management Protocol Server)
  - src/ — Source code files
  - test/ — Tests to the code
  - .editorconfig — Config file for supported code editors / IDEs
  - Makefile — Makefile
  - README.md — Readme file
- Instructions
  - Build project: `$ make`
  - Run tests: `$ make run_test`

FAP Controller + AP | Public API

- `int initializeFapController();`
- `int terminateFapController();`
- `int updateFapCoordinates();`

- Every function returns an `int`
  - return 0; → OK
  - return -1; → ERROR

N.B.: Refer to the API's documentation in the provided codebase.

FAP Controller + AP | Submission Instructions

- FAP Controller
  - .zip file with the source code
  - Note: Before zipping, please delete the executable file from the codebase by running the following command
    - `$ make clean`
- AP
  - List of Linux services used
  - List of configuration files created or modified
    - File's content
    - File's full path (e.g., "/etc/hostapd/hostapd.conf")

- By email to Eduardo Almeida [eduardo.n.almeida@inesctec.pt]