Secure and Reliable Communication Architecture

VanAssist - Interaktives, intelligentes System für autonome fernüberwachte Kleintransporter in der Paketlogistik

Institute of Reliable Embedded System and Communication Electronics (ivESK), Hochschule Offenburg

Gefördert durch:

Bundesministerium
für Verkehr und
digitale Infrastruktur

aufgrund eines Beschlusses
des Deutschen Bundestages

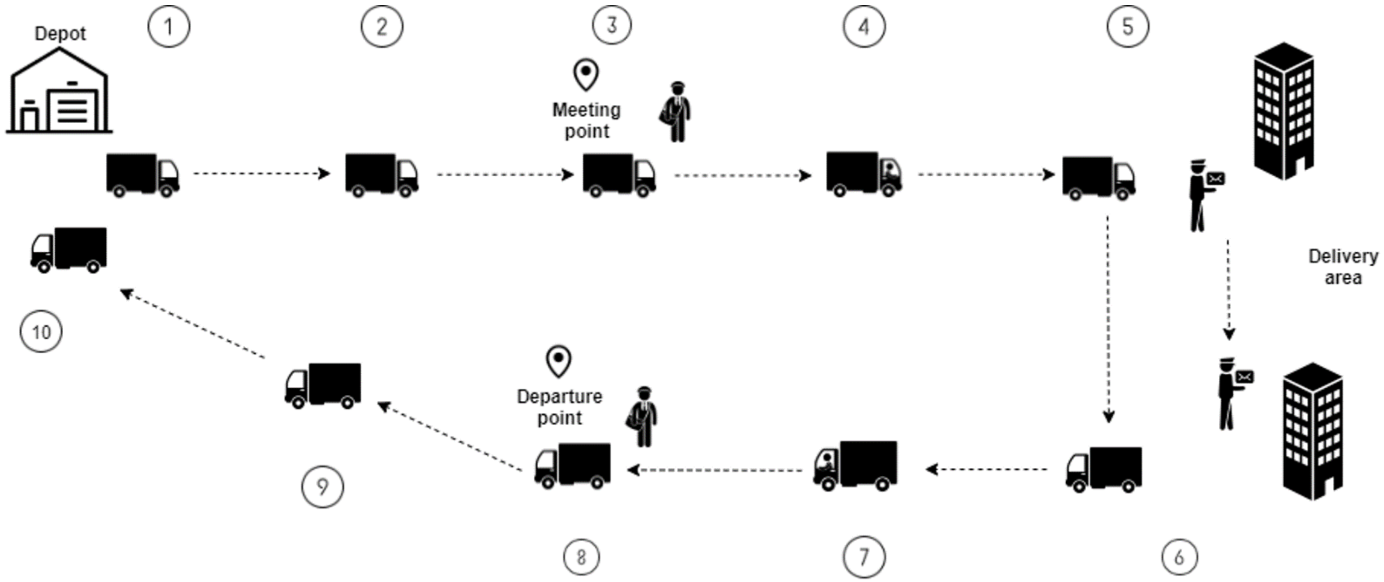
Agenda

- Introduction
    - System Use Case
    - Communication Requirements
    - Security Requirements
- Architecture of the Communication System
    - Architecture Description
    - Roles and Components
    - Primary and Secondary Communication
    - Interfaces and Protocols
- Security Design
    - Security Considerations
    - Security Domains
    - Security Implementation
- Interface Commands and Requests
- Implementation
- Integration
- Summary

Introduction

System Use Case

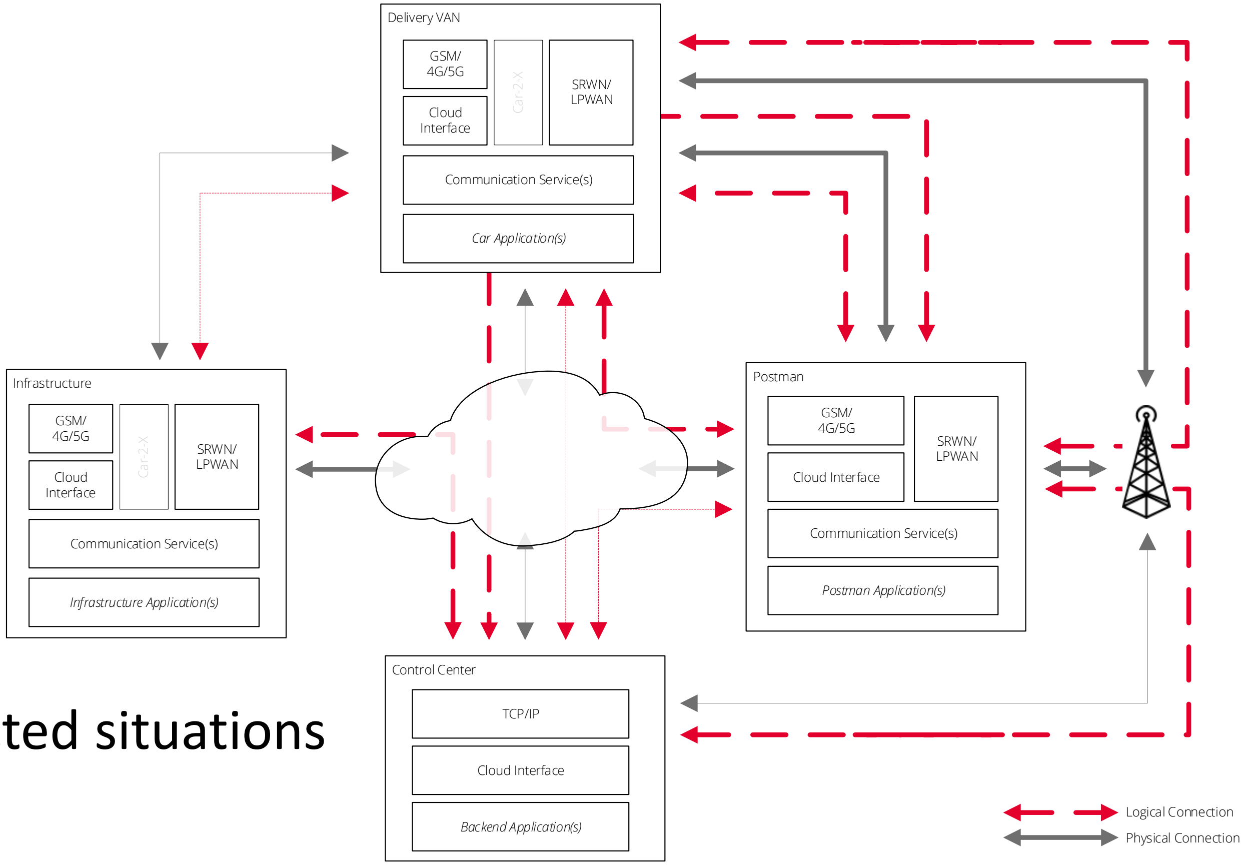
1. Van gets loaded in the Depot
2. Van drives to Meeting Point
3. Courier gets on the Van
4. Courier drives the Van to delivery area
5. Courier takes and delivers parcels
6. Courier calls the Van to the new location
7. Courier drives the van to Departure point
8. Courier leaves the van
9. Van drives to the Depot
10. Van gets unloaded in the depot and then sent to the parking



Introduction

Communication Requirements

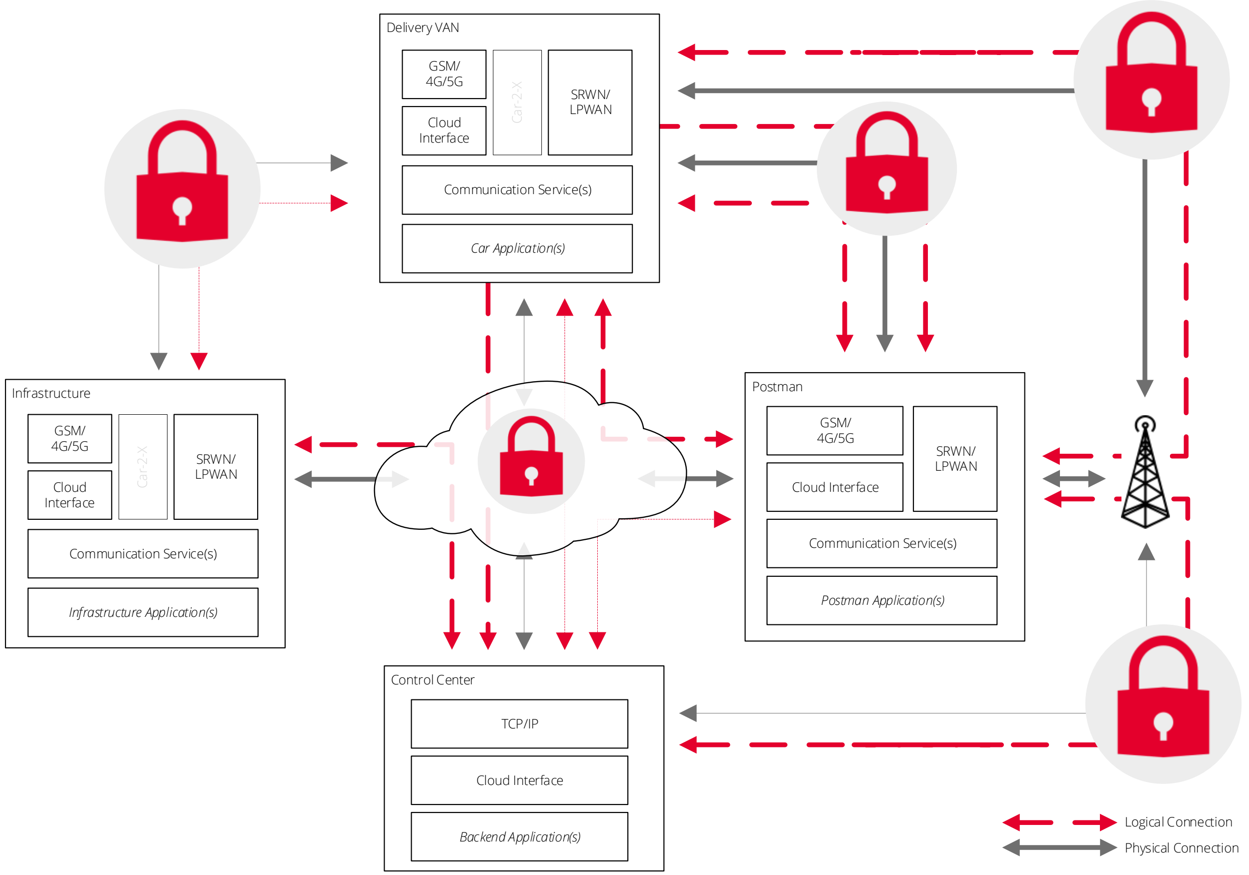
- VanAssist System Entities
  - Couriers
  - Vans
  - Backend
- Communication Needs
  - Courier gets updates from van
  - Courier sends commands to van
  - Backend monitors vans and handles unexpected situations
  - Couriers update FMS with delivery status



Introduction

Security Requirements

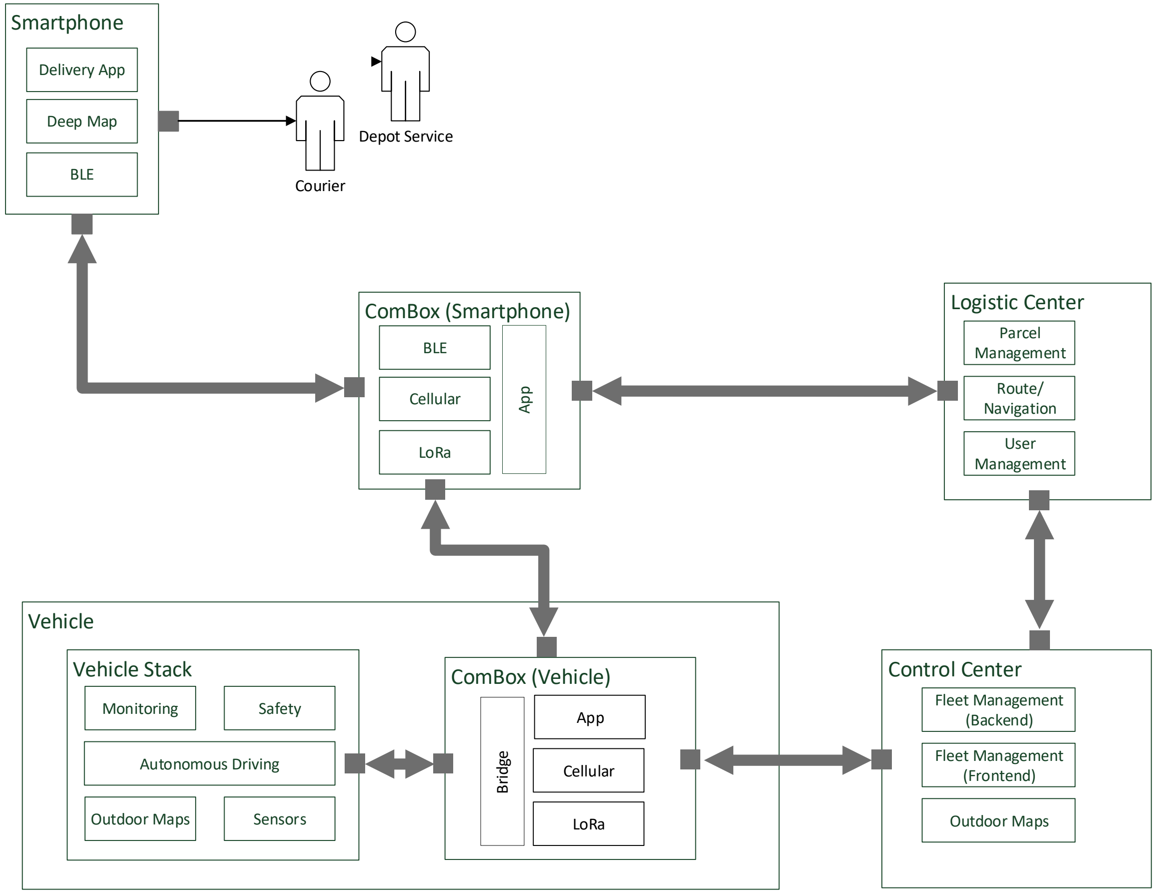
- Communication is on public network
- Possible attacks
  - Data leakage
  - Denial of Service (DoS)
  - Data tempering
  - Unauthorized access
  - Fake entity injection
  - …



Architecture of the Communication System

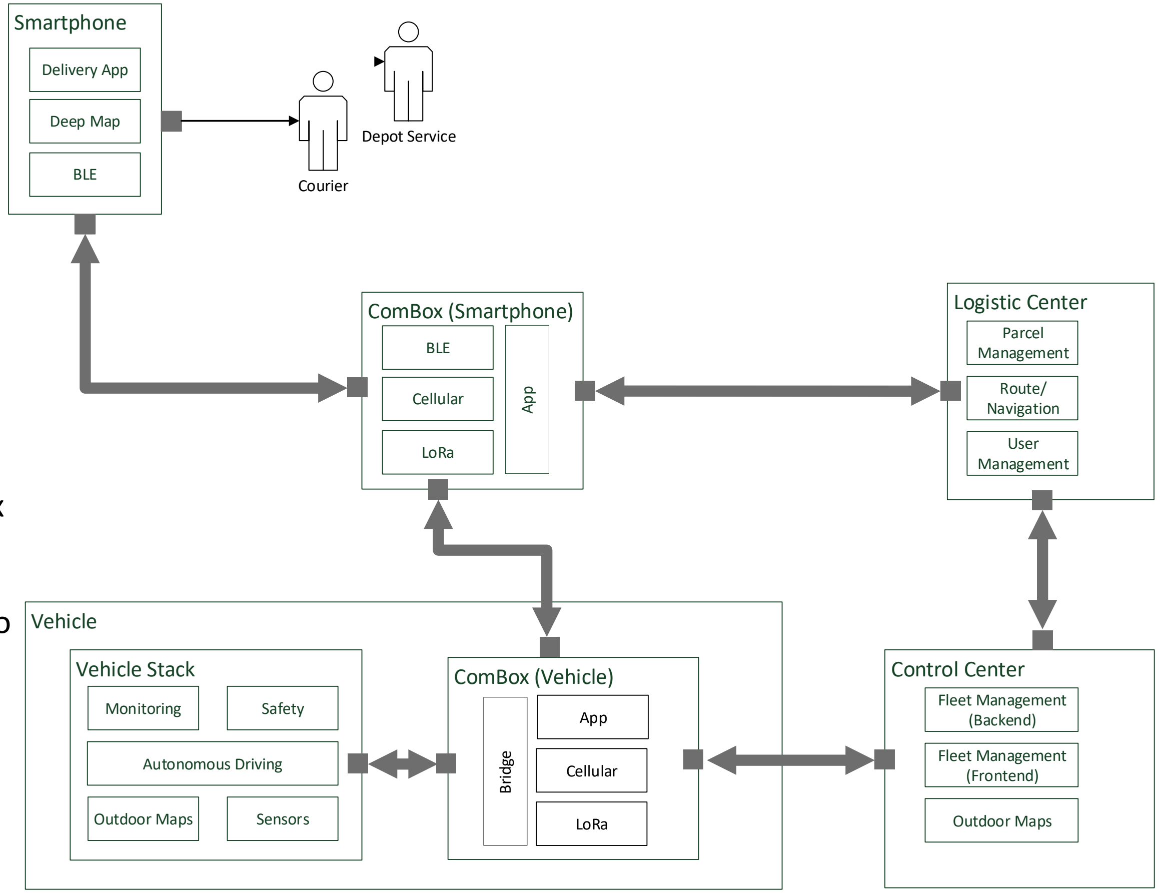
Architecture Description

- System users (courier, depot service) interact with the system via smartphone and a combox
- Vehicles also interact with the system via a combox
- A backend system acts as the backbone to the whole system and manages couriers-vehicles interactions



Architecture of the Communication System
Roles and Components

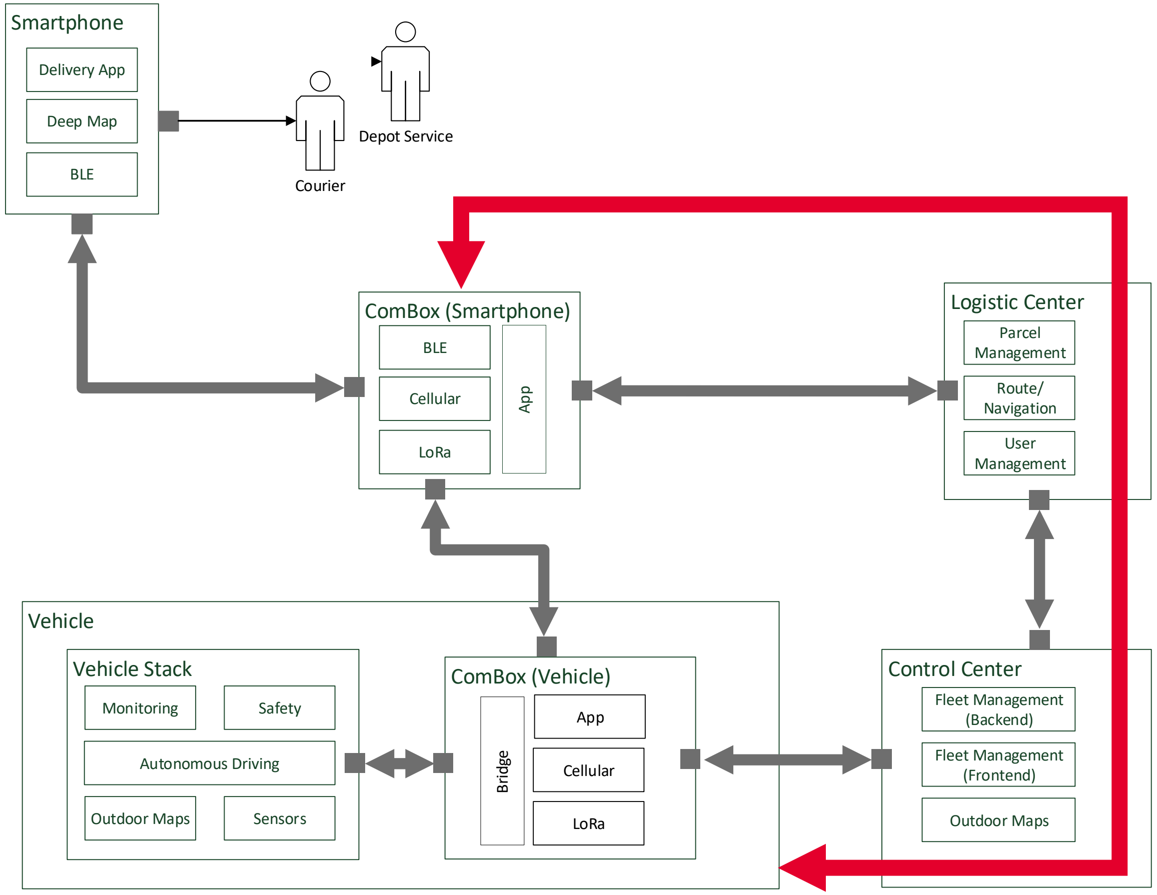
- Logistic Center
  - Manages and monitors couriers, parcels, navigation maps
- Control Center
  - Monitors and supervises delivery vans
- SBox
  - The interface between Courier's smartphone and Backend system (Logistic Center)
- Smartphone
  - Delivers van status to Courier and gets and relays Courier commands to SBox
- VBox
  - Gathers van status and relays them to Backend, relays received commands to van
- Vehicle Stack
  - Interacts with vehicle sensors and actuators to send status to VBox and deliver commands to vehicle



Architecture of the Communication System

Primary Communication

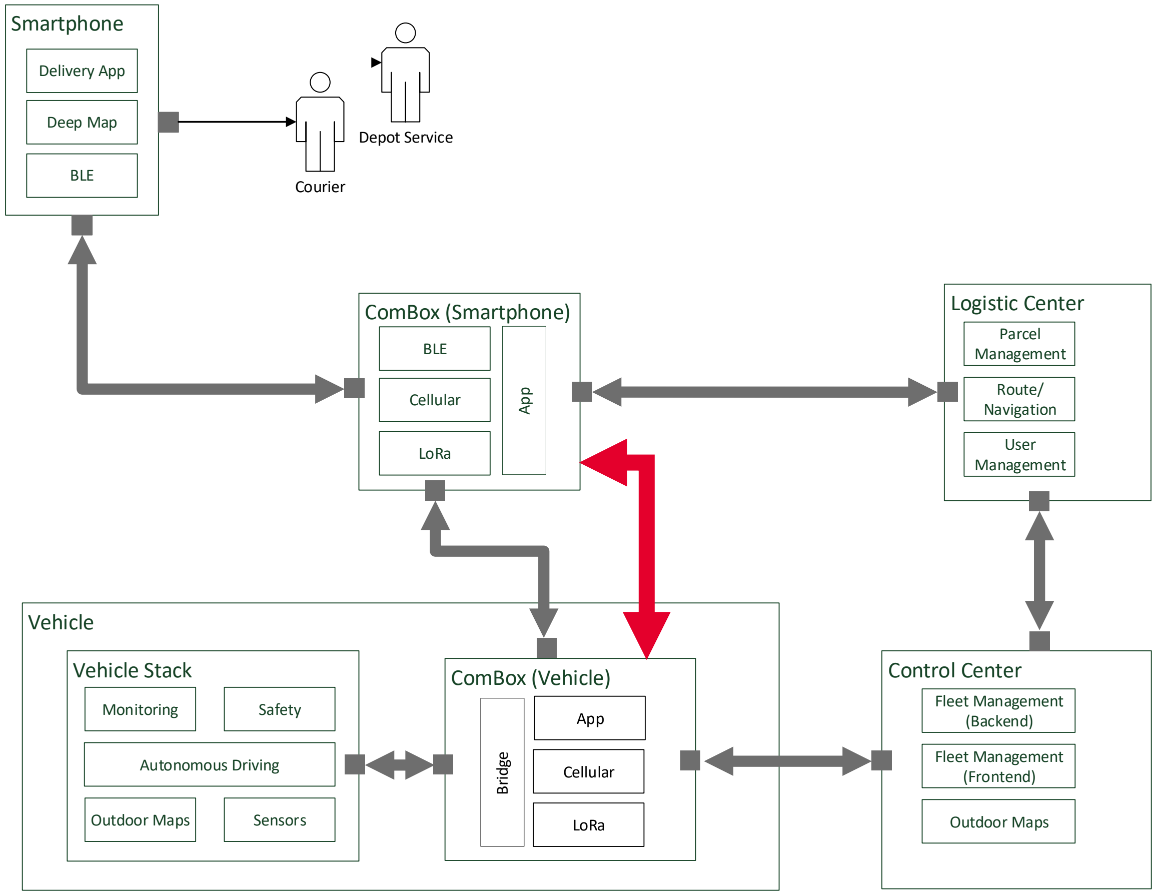
- Primary Communication is established via cellular network
- SBox from Courier domain and VBox from Vehicle domain have access to cellular network
- LC-CC connection is established via LAN/internet
- Every message between SBox and VBox is relayed through LC-CC



Architecture of the Communication System

Secondary Communication

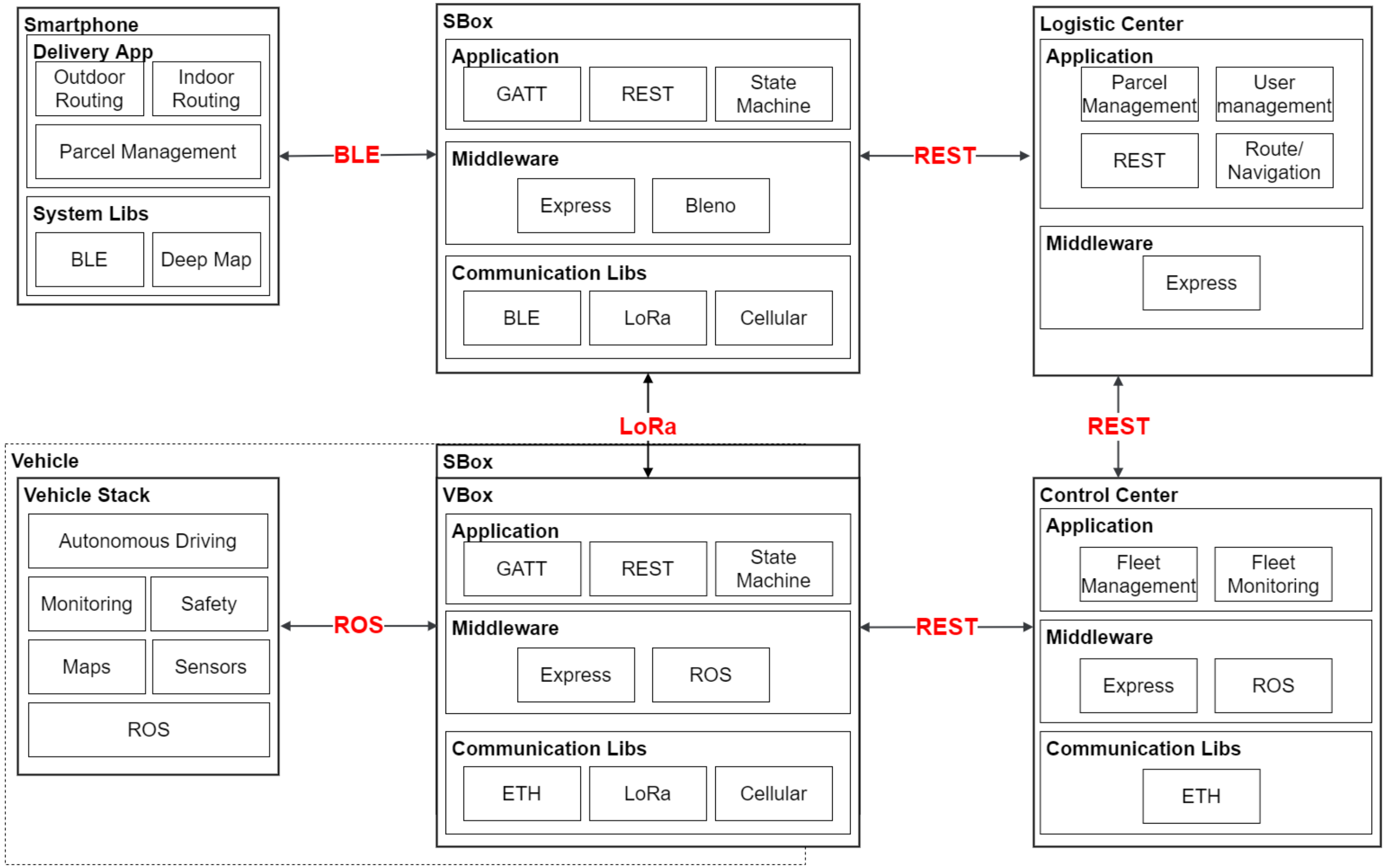
- Primary Communication requires cellular network coverage
- Cellular coverage might be [temporarily] unavailable in some areas
- To improve system reliability, a secondary communication path between SBox-VBox is designed
- Secondary communication protocol is LoRa



Architecture of the Communication System

Interfaces and Protocols

- **Bluetooth Low Energy (BLE)** between the App and the SBox
- **REST** for the communication to and within the backend systems
- **LoRaWAN** for the secondary communication
- **ROS** for the vehicle related functions



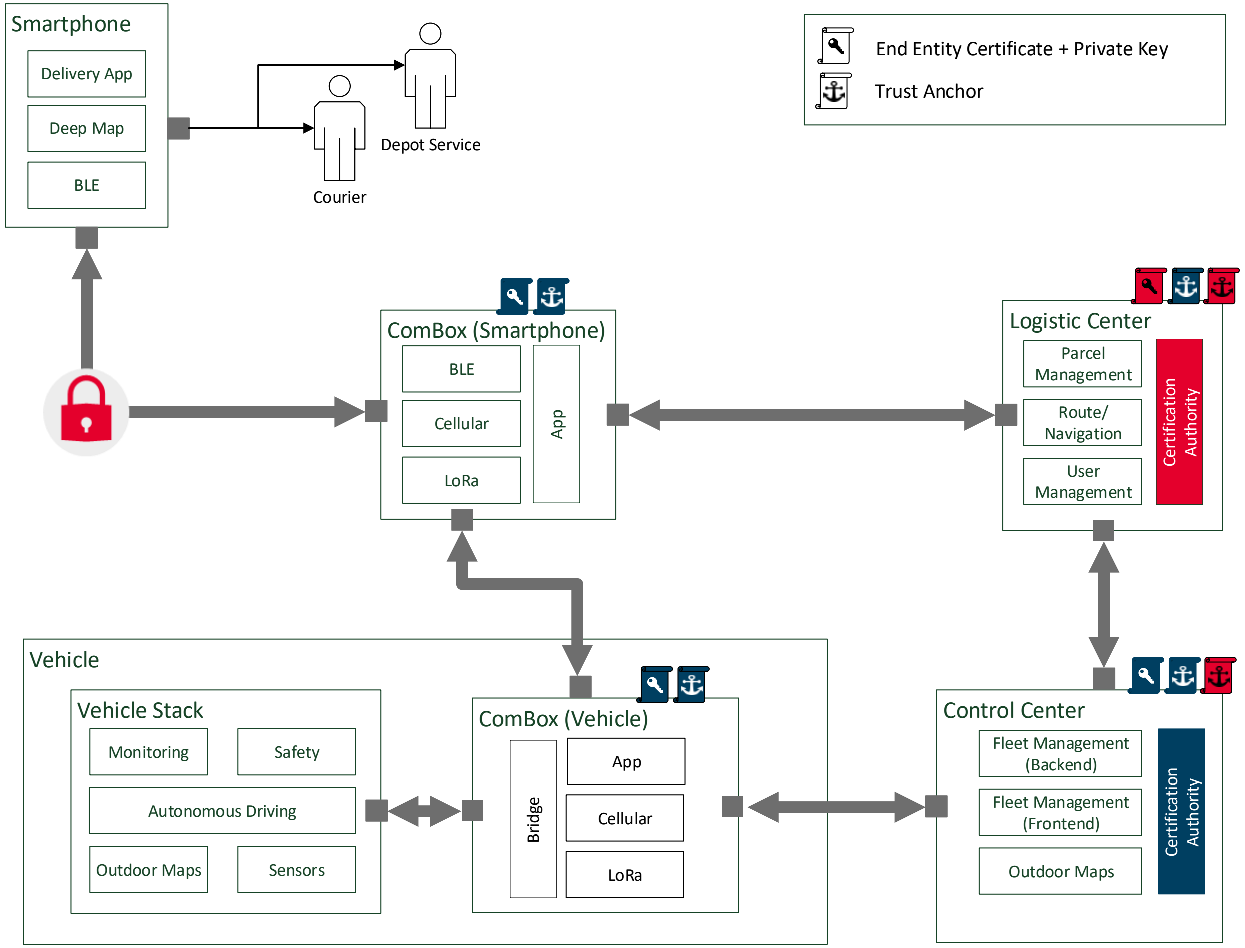
Security Design
Security Considerations

- Entity Authentication
  - Logistic Center – Control Center
  - SBox – Logistic Center
  - SBox – Control Center
  - VBox – Control Center
  - SBox – VBox (Secondary communication)
- Data Confidentiality and Integrity
  - All exchanged messages must be encrypted

Security Design

Security Domains

- Architecture is divided to 3 domains
  - Backend Domain
    - Logistic Center
    - Control Center
  - Courier Domain
    - SBox
    - Smartphone
  - Vehicle Domain
    - VBox
    - Vehicle Stack



Security Design

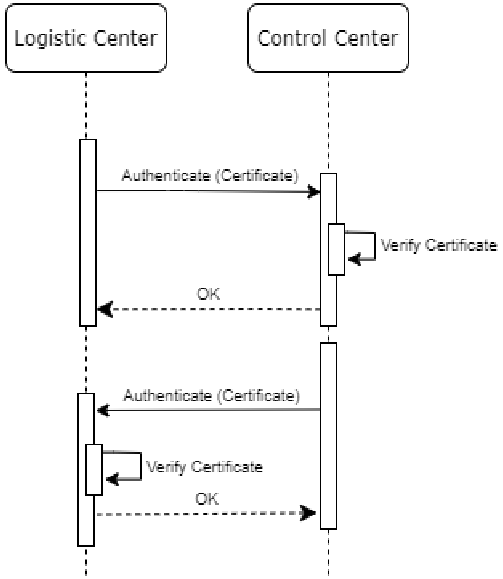
Security Implementation

- Logistic Center – Control Center: TLS certificates
- SBox – Logistic Center: Challenge – Response mechanism
- SBox – Control Center: TLS certificates
- VBox – Control Center: TLS certificates
- SBox – VBox: Pre-shared key via primary communication

Security Design

Security Implementation

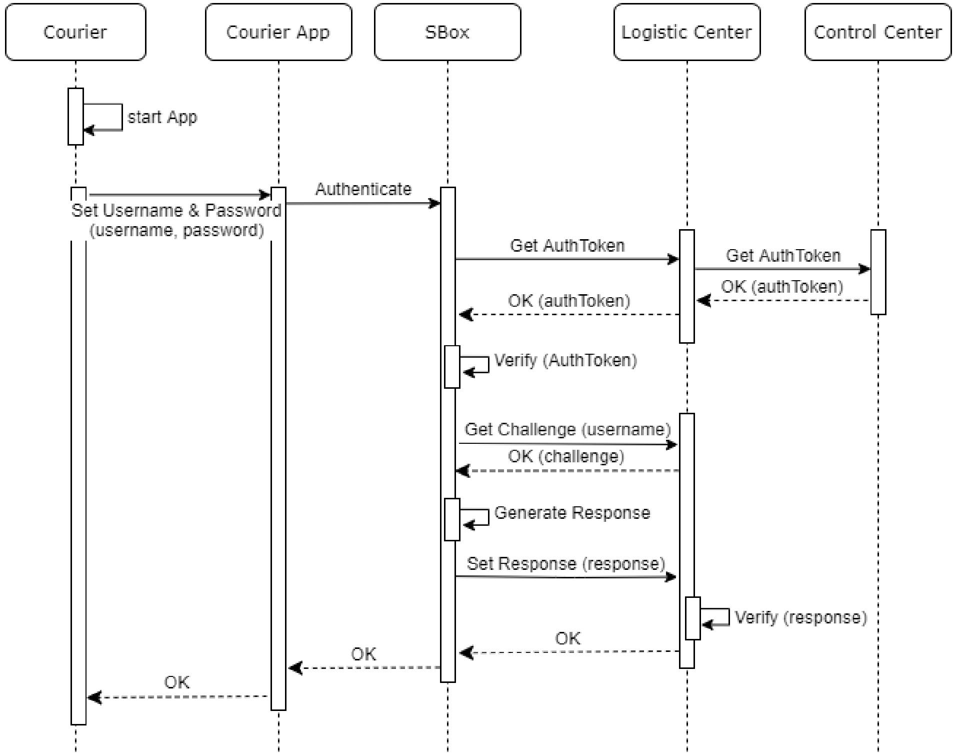
- Authentication between Logistic Center and Control Center
  - TLS-Based authentication
  - First, Control Center verifies Logistic Center certificate
  - Then, Logistic Center verifies Control Center certificate
  - After authentication, the session key is exchanged and the communication is encrypted using the session key



Security Design

Security Implementation

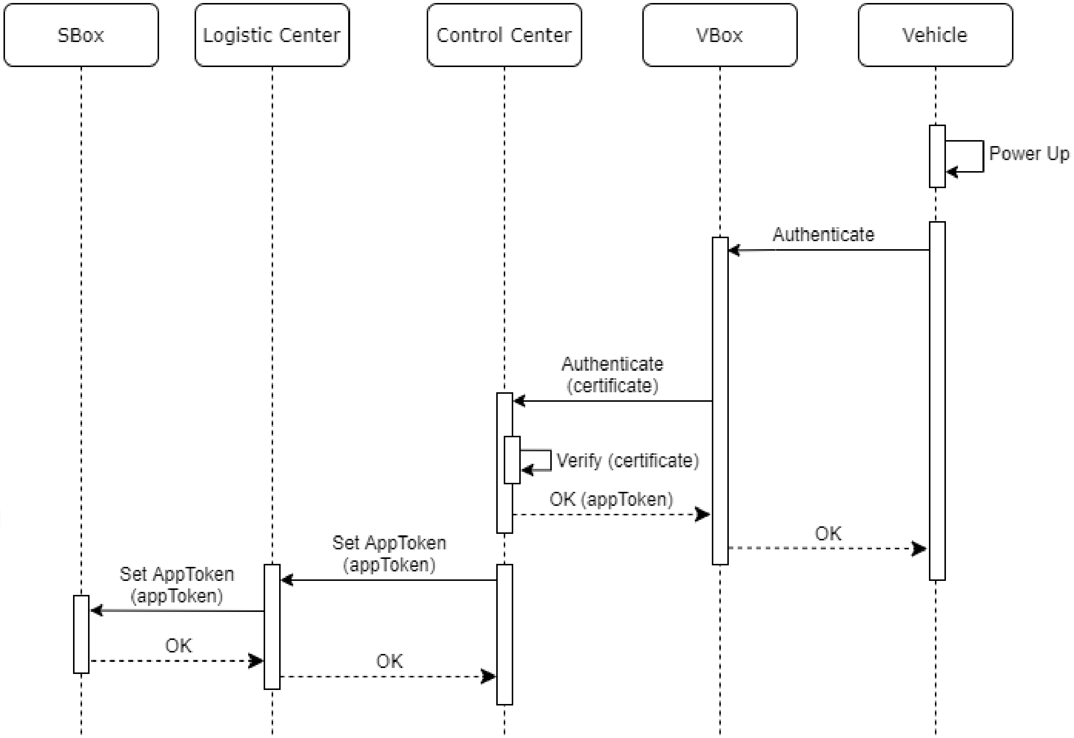
- Authentication SBox/Logistic Center
  - First, SBox requests the backend AuthToken
  - Logistic Center, retrieves backend AuthToken from Control Center and forwards it to SBox
  - After verifying AuthToken, SBox requests for authentication challenge from Logistic Center
  - Logistic Center generates and sends challenge for SBox
  - SBox generates and sends the corresponding response
  - Logistic Center verifies the response and authenticates SBox



Security Design

Security Implementation

- Authentication VBox/Control Center
  - After vehicle power-up, VBox sends its certificate to Control Center
  - Control Center verifies VBox certificate
  - If corresponding SBox is online, an AppToken is generated and sent to VBox and SBox



Security Commands and Requests

SBox and Logistic Center

SBox

URL	Method	Description
/api/v1/security/authtoken	GET	Retrieve authentication token from control center to verify logistic center
/api/v1/security/authchallenge	GET	Requesting for authentication challenge
/api/v1/security/authenticate	POST	Providing authentication response
/api/v1/fleet/vehicle/<vehicleid>/drivetopos	PUT	Send vehicle to a specific position
/api/v1/fleet/vehicle/<vehicleid>/status	PUT	change the status of the vehicle
/parcel/all	GET	Load all parcel data
/parcel/delivery/success	PUT	Confirm successful delivery
/parcel/delivery/failure	PUT	Confirm not successful delivery

Logistic Center

URL	Method	Description
api/v1/security/apptoken	POST	Send the application token to an SBOX
/api/v1/fleet/vehicle/<vehicleid>/drvsysstatus	PUT	Push the current driving system state of a vehicle
/api/v1/fleet/vehicle/<vehicleid>/currpos	PUT	Push the current position of a vehicle
/api/v1/fleet/vehicle/<vehicleid>/currtargetpos	PUT	Push the current target position of a vehicle
/api/v1/mgmt/connect	GET	Establish connection with control center
/api/v1/fleet/vehicle/<vehicleid>/drivetopos	PUT	Send vehicle to a specific position
/api/v1/fleet/vehicle/<vehicleid>/status	PUT	Set the status of the vehicle from operational point of view e.g. ready, error

Security Commands and Requests

Control Center and VBox

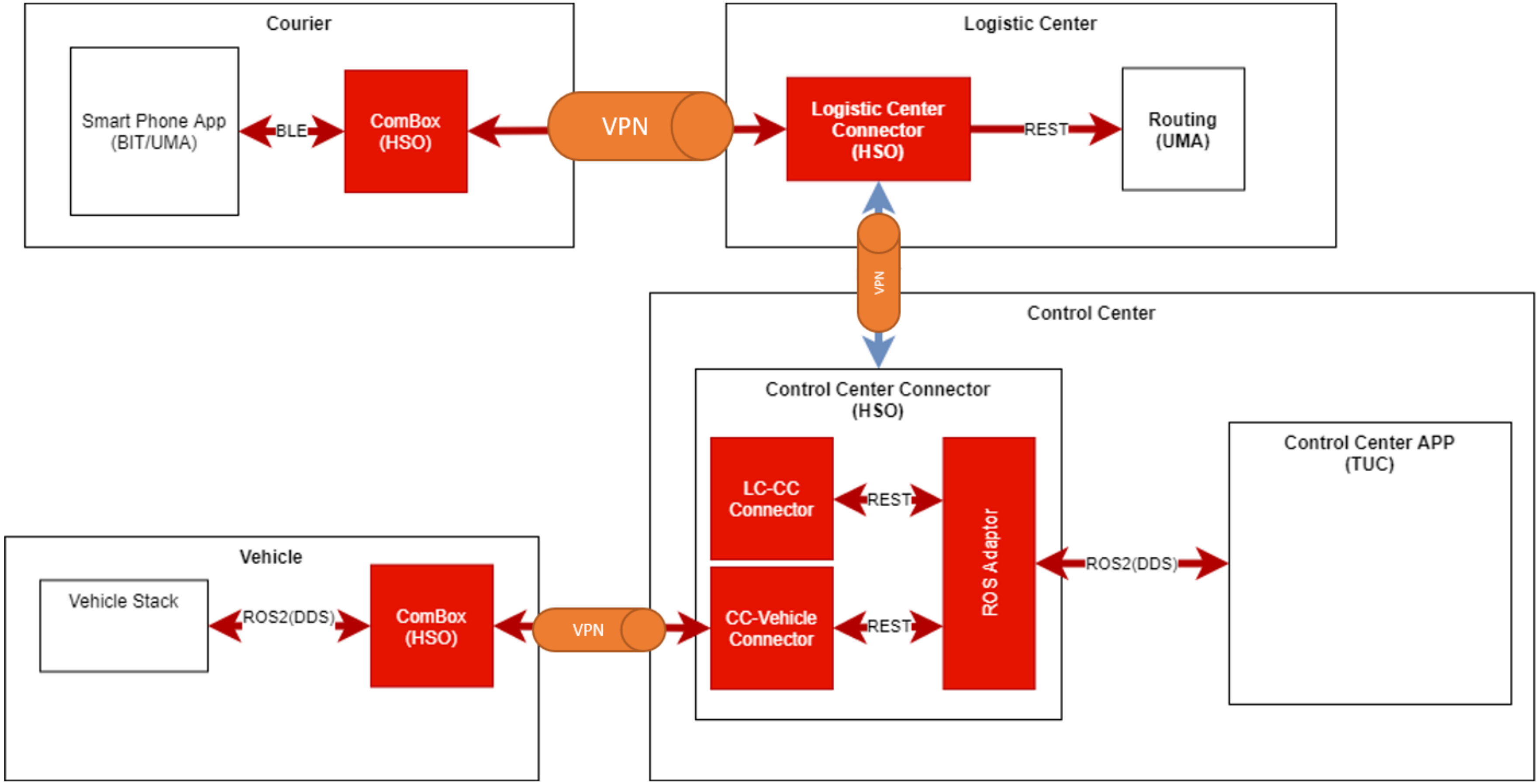
Control Center Connector

URL	Method	Description
/api/v1/fleet/vehicle/<vehicleid>/currpos	PUT	Send Current vehicle position
/api/v1/fleet/vehicle/<vehicleid>/currtargetpos	PUT	Send Current vehicle target position
/api/v1/fleet/vehicle/<vehicleid>/drvsysstatus	PUT	Push the current driving system state of a vehicle
api/v1/security/apptoken	POST	Send the application token to VBOX
api/v1/security/apptoken	DELETE	Delete the current application token from the VBOX
/api/v1/fleet/vehicle/drivetopos	PUT	Send vehicle to a specific position
/api/v1/fleet/vehicle/statectrl	PUT	change the status of the vehicle

VBox

URL	Method	Description
/api/v1/security/authenticate	POST	Delete the current application token from the VBOX
/api/v1/fleet/vehicle/<vehicleid>/currpos	PUT	Send Current vehicle position
/api/v1/fleet/vehicle/<vehicleid>/currtargetpos	PUT	Send Current vehicle target
/api/v1/fleet/vehicle/<vehicleid>/drvsysstatus	PUT	Push the current driving system state of a vehicle
/api/v1/fleet/vehicle/<vehicleid>/sensor/cam/<id>	PUT	Send camera data to control center
/api/v1/fleet/vehicle/<vehicleid>/sensor/lidar/<id>	PUT	Send lidar data to control center

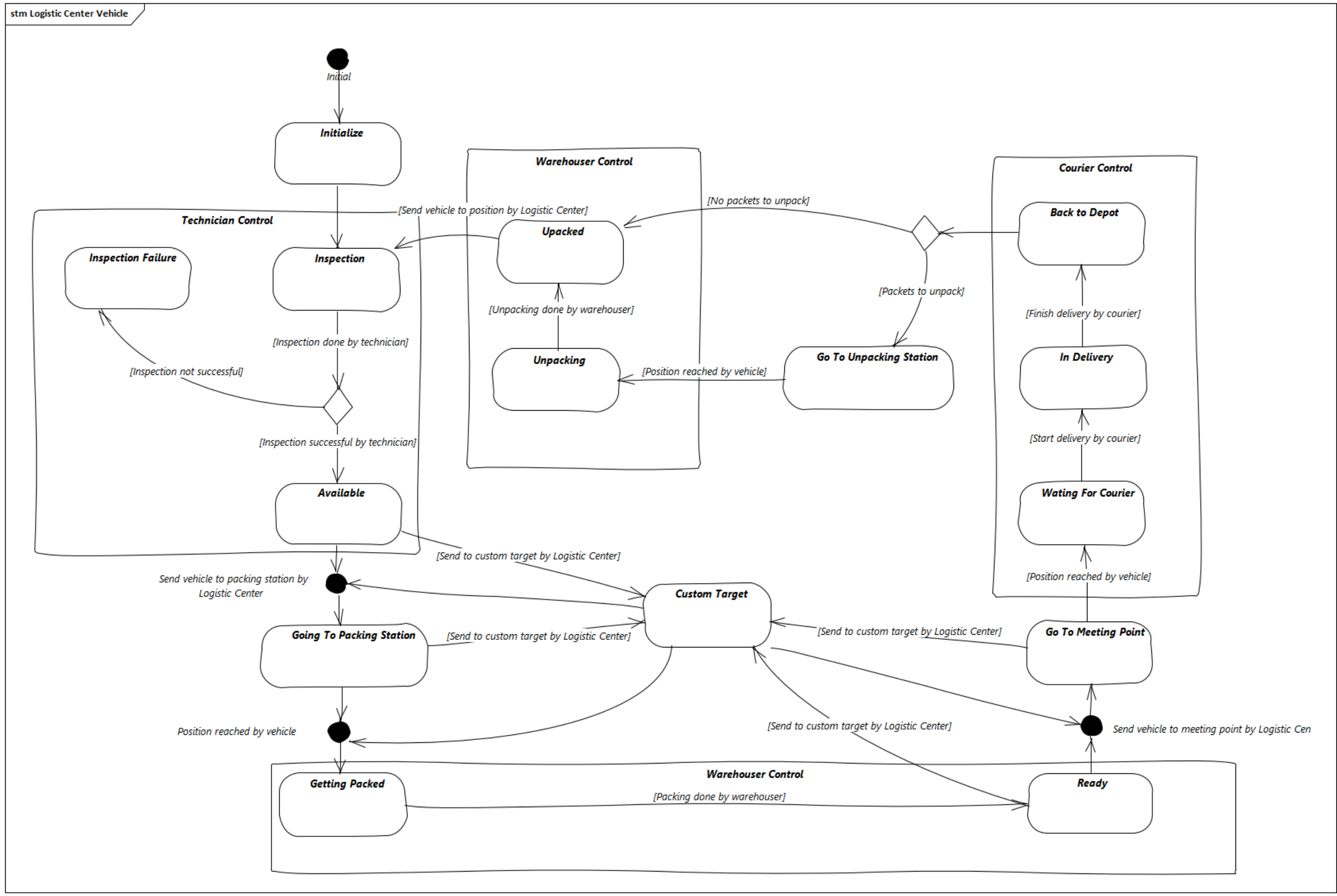
Implementation



Implementation

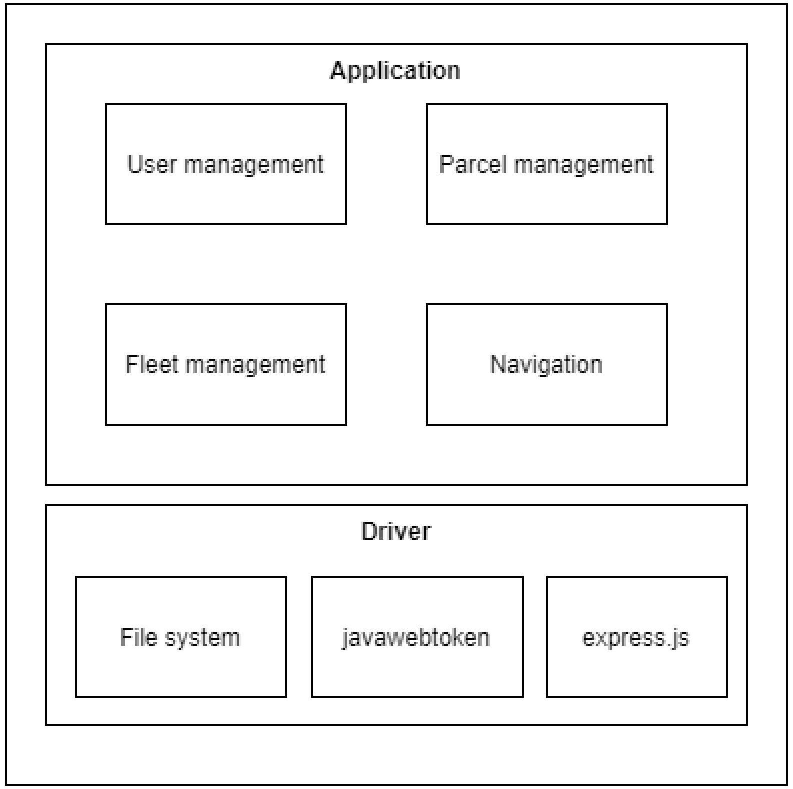
Backend Implementation (Logistic Center)

- Backend State Machine
  - After start-up system goes to initialize state
  - Then it goes to Technician‘s control
  - Afterwards, it goes to warehouse for loading
  - Then it goes to the meeting point and delivery starts
  - In the end, it goes back to warehouse for unloading undelivered parcels
  - In the end, it goes to parking and waits for the next round


stm Logistic Center Vehicle
Initial
Initialize
Technician Control
Inspection Failure
Inspection
[Inspection done by technician]
[Inspection not successful]
[Inspection successful by technician]
Available
Warehouser Control
[Send vehicle to position by Logistic Center]
Upacked
[Unpacking done by warehouser]
Unpacking
[Position reached by vehicle]
[No packets to unpack]
[Packets to unpack]
Go To Unpacking Station
Courier Control
Back to Depot
[Finish delivery by courier]
In Delivery
[Start delivery by courier]
Wating For Courier
[Position reached by vehicle]
[Send to custom target by Logistic Center]
Send vehicle to packing station by Logistic Center
Custom Target
Going To Packing Station
[Send to custom target by Logistic Center]
[Send to custom target by Logistic Center]
Go To Meeting Point
Position reached by vehicle
[Send to custom target by Logistic Center]
Send vehicle to meeting point by Logistic Cen
Warehouser Control
Getting Packed
[Packing done by warehouser]
Ready

Implementation

Backend Implementation



Application

User management

Parcel management

Fleet management

Navigation

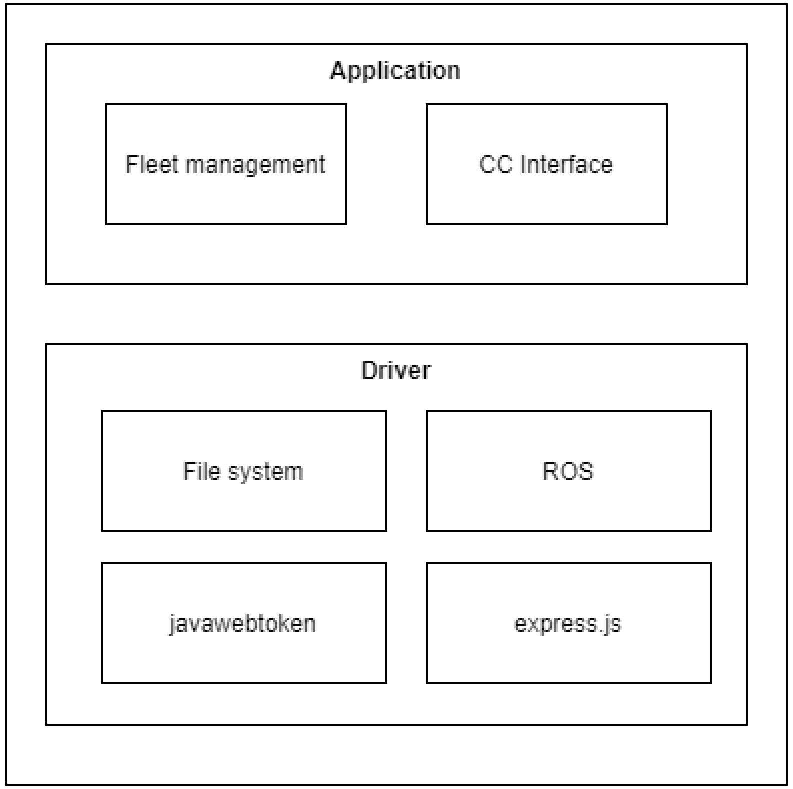
Driver

File system

javawebtoken

express.js

Logistic Center



Application

Fleet management

CC Interface

Driver

File system

ROS

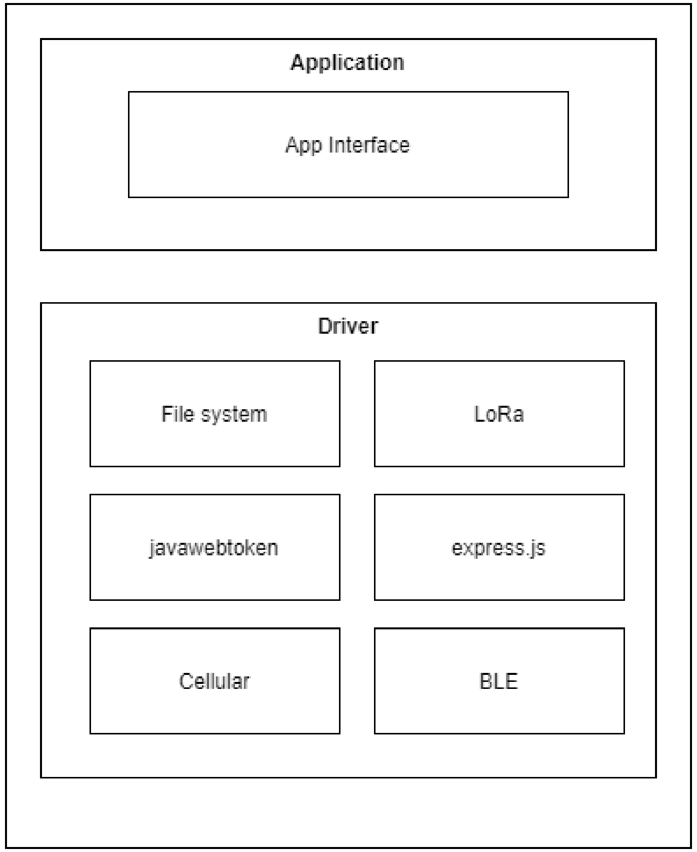
javawebtoken

express.js

Control Center Connector

Implementation

ComBoxes Implementation



Application

App Interface

Driver

File system

LoRa

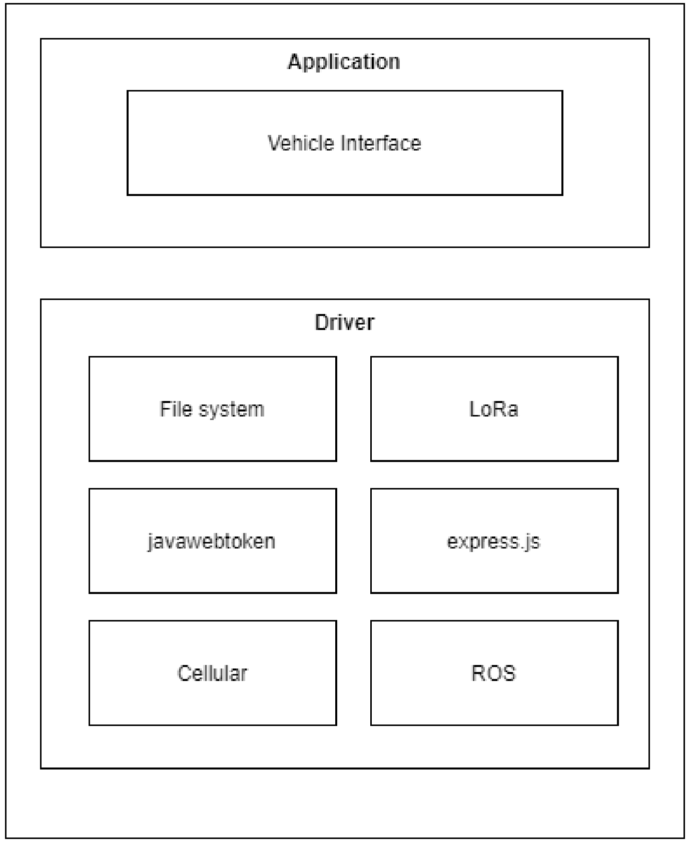
javawebtoken

express.js

Cellular

BLE

SBox



Application

Vehicle Interface

Driver

File system

LoRa

javawebtoken

express.js

Cellular

ROS

VBox

Integration

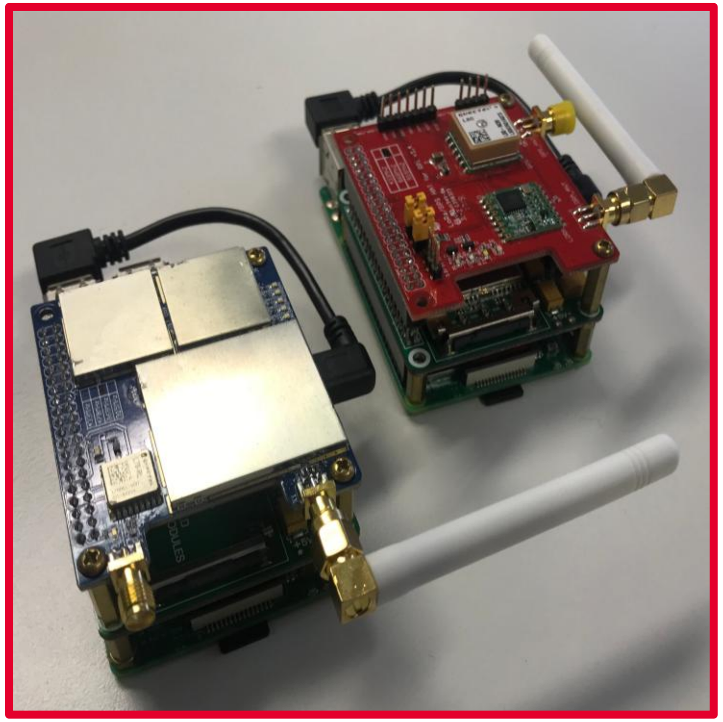
Integration Steps

- Individual Units
  - Implementation and testing of system units
- REST Interfaces
  - Test the REST interface between SBox, Logistic Center, Control Center, Vbox
  - Emulate vehicle with simulator
- ROS Interfaces
  - Test the ROS interface between VBox-Vehicle and inside Control Center
- LoRa Interface
  - Test the LoRa secondary interface between SBox and VBox
- Final Test
  - Test all the interfaces with backend system and vehicle enabled

Integration

ComBoxes

- Common Base Platform (Raspberry Pi 3)
- Common Cellular-Shield
- Separate LoRa-Shields for VBox and SBox
  - LoRa Device (SBox)
  - LoRa Concentrator (VBox)
- Common Operating System (Ubuntu)
- Common Software Modules (e.g. Node-js)



Integration

Logistic Center – Control Center Connector

- Hardware Platform: Industrial PC
- Internet Connection via Ethernet
- Linux Operating System (Ubuntu)
- Common Software Modules (e.g. Node-js)

Summary

- In the first step, communication and security requirements of the system have been specified
- Then, according to the requirements, system architecture has been designed and system entities are defined
- Primary and secondary communications have been desined to improve system reliability
- The system has been divided to several security domains
- Different authentication and encryption mechanisms have been employed to ensure system security
- The system has been implemented and tested step by step from unit test to comprehensive test of the integrated system