Documentation Ev-Charging

Course: Distributed System

Students: Emily Apitzsch

Students: Johanna Helena Baarssen

Date: 02.11.2025

Universidad de Alicante

Documentation – EvCharging Project (SD 2025/26)

Authors: Emily Apitzsch, Johanna Helena Baarssen
Date: 02.11.2025

1. Project Description

The objective of this project is to design and implement a distributed simulation of an Electric Vehicle (EV) charging network, following real EV charging system behavior and distributed systems principles. The system architecture is inspired by industry-standard infrastructures such as OCPP (Open Charge Point Protocol), combining socket-based real-time control with event-streaming telemetry.

This project was developed for the Distributed Systems course and shows concepts such as:

- Decentralized computing
- Asynchronous communication
- Heartbeat and fault detection
- High-availability and reconnection logic
- Event streaming (Apache Kafka)
- Persistent state and system recovery
- Multi-component orchestration
-

The solution models all main actors in an EV charging project:

- Central Control Server
- Charging Point (Engine)
- Charging Point Monitor
- Driver Application

2. Components Description

2.1 EV_Central

The Central Server is the core control logic. It:

- Listens for socket connections from CPs and Drivers
- Registers charging points and stores them in SQLite
- Assigns charging points to drivers on request
- Stores and updates CP status
- Displays real-time dashboard state (colors, kW, €)
- Receives telemetry & alerts via Kafka consumers
- Handles failures and transitions:

The Central never stops running and is available at all times.

2.2 CP Engine (EV_CP_E)

Simulates the charging hardware and business logic:

- Connects to Central and registers its location and price
- Simulates charging process (kWh, EUR accumulation)
- Sends telemetry every second to Kafka (CHARGING, FINISHED)
- Reconnects to Central automatically if connection fails

It represents the operational part of the charger.

2.3 CP Monitor (EV_CP_M)

The Monitor module simulates the internal diagnostics board for each charging station:

- Maintains a TCP connection to the Engine
- Sends heartbeat messages every second
- Sends alerts to Central via Kafka
- Detects network failures and engine crashes

It ensures fault tolerance and real-time diagnostics, just like real EV stations.

2.4 Driver Application (EV_Driver)

Simulates EV users requesting charging sessions:

- Connects to Central and authenticates
- Handles Central messages (START, STOP)
- Waits between services to simulate real users

3. Communication Protocol

For socket communication, a custom frame protocol is used:

```
<STX><PAYLOAD><ETX><LRC>
```

Example message:

```
02 START_REQ#DRIVER01 03 5A
```

4. Persistence & Recovery

The system persists charging-point data in SQLite:

- IDs
- Location
- Price
- Status

On restart:

- CENTRAL loads CP registry
- Devices reconnect automatically
- Disconnected CPs display as grey / DISCONNECTED until heartbeat resumes

This ensures state continuity

5. Deployment Guide:

This section explains how to deploy and run the EV Charging Distributed System on multiple machines.

Install Environment:

macOS / Linux:

```
cd DiSystems
python3 -m venv venv
source venv/bin/activate
pip install -r common/requirements.txt
```

Windows:

```
cd DiSystems
python -m venv venv
.\venv\Scripts\activate
pip install -r common\requirements.txt
```

Start Kafka:

```
docker compose up -d
```

Start Central:

```
cd DiSystems
source venv/bin/activate
python -m central.EV_Central 9002
```

Start CP:

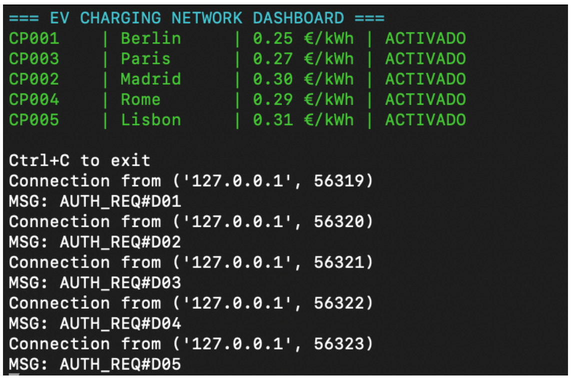
```
cd DiSystems
venv\Scripts\activate
python -m cp.EV_CP_E CP001 Berlin 0.25 9101 --central-ip <CENTRAL-IP>
python -m cp.EV_CP_M CP001 9101 --central-ip <CENTRAL-IP>
```

Start Driver:

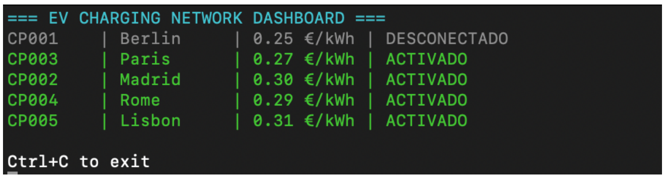
```
python -m driver.EV_Driver --central-ip <CENTRAL-IP> --central-port 9002 --driver-id D01
```

6. Screenshots:

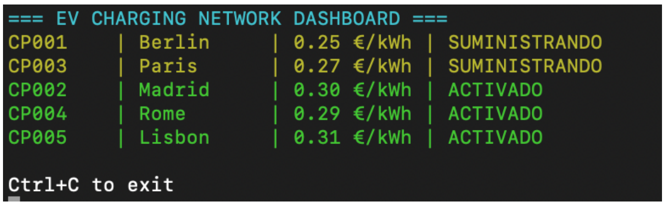
Central:



```
=== EV CHARGING NETWORK DASHBOARD ===
CP001     | Berlin      | 0.25 €/kWh | ACTIVADO
CP003     | Paris       | 0.27 €/kWh | ACTIVADO
CP002     | Madrid      | 0.30 €/kWh | ACTIVADO
CP004     | Rome        | 0.29 €/kWh | ACTIVADO
CP005     | Lisbon      | 0.31 €/kWh | ACTIVADO

Ctrl+C to exit
Connection from ('127.0.0.1', 56319)
MSG: AUTH_REQ#D01
Connection from ('127.0.0.1', 56320)
MSG: AUTH_REQ#D02
Connection from ('127.0.0.1', 56321)
MSG: AUTH_REQ#D03
Connection from ('127.0.0.1', 56322)
MSG: AUTH_REQ#D04
Connection from ('127.0.0.1', 56323)
MSG: AUTH_REQ#D05
```

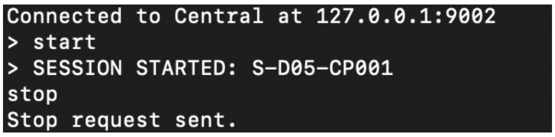


```
=== EV CHARGING NETWORK DASHBOARD ===
CP001     | Berlin      | 0.25 €/kWh | DESCONECTADO
CP003     | Paris       | 0.27 €/kWh | ACTIVADO
CP002     | Madrid      | 0.30 €/kWh | ACTIVADO
CP004     | Rome        | 0.29 €/kWh | ACTIVADO
CP005     | Lisbon      | 0.31 €/kWh | ACTIVADO

Ctrl+C to exit
```

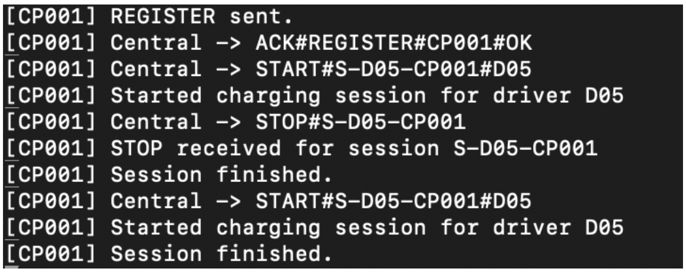


```
=== EV CHARGING NETWORK DASHBOARD ===
CP001     | Berlin      | 0.25 €/kWh | SUMINISTRANDO
CP003     | Paris       | 0.27 €/kWh | SUMINISTRANDO
CP002     | Madrid      | 0.30 €/kWh | ACTIVADO
CP004     | Rome        | 0.29 €/kWh | ACTIVADO
CP005     | Lisbon      | 0.31 €/kWh | ACTIVADO

Ctrl+C to exit
```

Driver:



```
Connected to Central at 127.0.0.1:9002
> start
> SESSION STARTED: S-D05-CP001
stop
Stop request sent.
```



```
[CP001] REGISTER sent.
[CP001] Central -> ACK#REGISTER#CP001#OK
[CP001] Central -> START#S-D05-CP001#D05
[CP001] Started charging session for driver D05
[CP001] Central -> STOP#S-D05-CP001
[CP001] STOP received for session S-D05-CP001
[CP001] Session finished.
[CP001] Central -> START#S-D05-CP001#D05
[CP001] Started charging session for driver D05
[CP001] Session finished.
```