KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree project in Electrical Engineering and Computer Science

Second cycle, 30 credits

Development of a smart charging management system for heavy-duty trucks

XIAOYING SUN

Stockholm, Sweden 2022

Author

Xiaoying Sun <xsu@kth.se>
MSc Software Engineering of Distributed Systems
KTH Royal Institute of Technology

Place for Project

KTH, Stockholm, Sweden
Scania, Södertälje, Sweden

Examiner

Ming Xiao, <mingx@kth.se>
KTH Royal Institute of Technology

Supervisors

Hampus Andersson <hampus.andersson@scania.com>

Scania Group

Zhibo Pang <zhibo@kth.se>

KTH Royal Institute of Technology

Abstract

This paper reviews the Open Charge Point Protocol (OCPP) and implements a Charging Station Management System (CSMS) targeting heavy-duty trucks.

The new technique proposed in this paper is designed to maximize Electric vehicle (EV) owner benefits by charging at a low cost, and also the electric utility benefits (operating the system within the acceptable limits) by proper choice of electricity tariff structure. EV owners can be motivated to charge at off-peak hours which have low electricity prices and stop charging at peak hours which have high electricity prices.

Keywords

Smart Charging, Electric vehicle, OCPP, OCPP 1.6, Charging optimization

Abstract

Detta dokument granskar Open Charge Point Protocol (OCPP) och implementerar en Charging Station Management System (CSMS) inriktat på tunga lastbilar.

Den nya tekniken som föreslås i detta dokument är utformad för att maximera fördelarna för ägare av elfordon (EV) genom att ladda till en låg kostnad, och även fördelarna med elnätet (drift av systemet inom acceptabla gränser) genom korrekt val av elprisstruktur. Elbilsägare kan motiveras att ladda under lågtrafik som har låga elpriser och sluta ladda under rusningstid som har höga elpriser.

Nyckelord

Smart Charging, Electric vehicle, OCPP, OCPP 1.6, Charging optimization

Acknowledgements

This study would not have been possible without the guidance and support from several people. First of all, I would like to express my sincere gratitude to my supervisor, Andersson Hampus, for his willingness to share his expertise and useful suggestions on my thesis. It is under his help that I have the privilege to have online meetings with professionals from Scania IT department and Kempower engineer.

I would alos like to express my gratitude to Dr. Zhibo Pang and Ming Xiao, my supervisor and examiner at KTH, for their valuable input during discussions at our meetings. Your perspective and experience from both academia and business have been valuable for me in this thesis.

Finally, I would love to express my thankfulness to my beloved friends, Pavel Gordon and Christine Arkbo, for their direct and generous help, special thanks should go to them for their contribution to the project and the thesis.

Last but not least, a special thanks to my family and corridor friends for their uncompromising support in my everyday life during these years. I would not graduate without your support.

Acronyms

AWS Amazon Web Service

AI Artificial Intelligence

ANN Artificial Neural Network

API Application Programming Interface

URI Uniform Resource Identifier

CSMS Charging Station Management System

CS Charging Station

CP Charging Point

EV Electric vehicle

BEV Battery Electric Vehicles

PHEV Plug-in Hybrid Electric Vehicles

ICEV Internal combustion engine vehicle

EVSE Electric Vehicle Supply Equipment

OCPP Open Charge Point Protocol

OCA Open Charging Alliance

SoC State of Charge

MA Moving Average

SMA Simple Moving Average

EMA Exponential Moving Average

SVM Support Vector Machine

V2G Vehicle to Grid

KF Kalman Filter

UKF Unscented Kalman Filter

TAGS Threshold admission and greedy scheduling

RDS Relational Database Service

EC2 Elastic Compute Cloud

VPC Virtual Private Cloud

DB Database

UML Unified Modeling Language

MVP Minimum Viable Product

V$_{DC}$ Volts Direct Current

TCP Transmission Control Protocol

GA Greedy Algorithm

APE Absolute Percentage Error

MAPE Mean Absolute Percentage Error

PDU Professional Development Units

Contents

Chapter 1

Introduction

Transitioning the global economy to sustainable development is one of the challenges in the coming decades. The Swedish government has an ambitious goal that 100% of Swedish electricity consumption will come from renewable sources by 2040. [18].

EV have advantages over conventional Internal combustion engine vehicle (ICEV) in terms of both energy conservation and reducing dependence on traditional fossil fuels. Meanwhile, the improvement in battery technology has made Scania Battery Electric Vehicles (BEV) and Plug-in Hybrid Electric Vehicles (PHEV) far more usable, not only for commuting short distances but also inter-city travel. As a consequence, the number of EVs have grown and the need for an intelligent charging management system is imminent.

Because innumerable EVs (domestic car, heavy-duty vehicles, etc.) will be charged from the electric power grid, both government and companies are crying out for new techniques for the optimal charging transaction of EVs.

There are three main types of EV charging; uncontrolled charging, delayed charging and smart charging. Uncontrolled charging means that EVs start charging at the instant of connecting with the charger. It is the type of EV charging that is used nowadays, in which case, EV charging usually occurs at peak load hours which leads to severe grid impacts [6]. This paper attempts to propose a smart charging algorithm to use the electric system in a more optimal and efficient way, by controlling of charging start time and charging rates of EVs.

1.1 Goals and Objectives

The objective of this paper is to charge EVs at a lower cost while meeting basic requirements of the customers. It is expected to This paper expects to deliver the following results.

- A detailed discussion on various smart charging objectives involving central smart charging and electricity price is provided.
- Two aspects of EV smart charging configuration are reviewed: battery capacity estimation and CSMS scheduling.
- A practical back-end charging management system implementation along with the major results.
- Discussions, research gaps and recommendations for future work related to EV smart charging are also included.

1.2 Problems

1.2.1 Problem Statement

There are environmental and economical factors that drive the electrification of the heavy duty vehicle industry and sales are growing fast. Electric heavy trucks have high voltage battery systems (around 800 V) with storage capacity of several hundred kWh and consume several hundreds megawatt per year. There are significant savings to be made by optimizing the charging of such vehicles by taking different aspects into account, for example the electricity price or the maximum grid capacity.

At present neither the vehicle nor the charger provide much optimization of the charging transaction. Essentially the charging starts when the charge cable is connected and goes at maximum power until the battery is full. Many chargers are connected to the internet and can be controlled by a back-end with a public protocol – OCPP.

1.2.2 Research Questions

- What are the renovations and limitations of contemporary EV smart charging solutions?

- What is the applicable way to infer the required energy and time from the raw data of the charging station?
- What are the most effective charging strategies in combination with the hourly electricity price?
- How well does the proposed method perform in maximizing customer benefits?
- What are the pioneering points and risks of the proposed method?

1.3 Ethics, Risks and Sustainability

Challenges

The challenges faced in this paper consist of two parts, theoretical and practical.

In theory, OCPP 1.6 is a complete and comprehensive contract, not all of whose provisions are relevant to this paper. How to sift through and familiarize with pertinent operations in limited time. The challenge is also that all explorations of future possibilities have to be placed in the context of existing contracts (OCPP 1.6 and ISO 15118) and defined frameworks (Test Tool OCPP 1.6).

Practical risks exist in data collection and third-party intellectual property protection. To collect data samples, the author is in need of the help from the supervisor to book a low-State of Charge (SoC) heavy-duty truck from Scania®in each iteration. A rash test (remote start or remote stop charging station) without a reservation will bring disorder and loss to normal production activities. What is more, low SoC battery ensures enough space to test remote switching and smart charging solutions, so sometimes we need to drive the truck around the training ground with no purpose.

On the other hand, the author is also responsible to check with Kempower®if it is OK to have snapshots of their application in this paper. It might be sensitive as it could provide its competitors with insights into ChargEye.

Sustainability

A smart charging solution among the smart grid, connectors, and EVs can bring various benefits to all parties involved, such as improved reliability and safety for the smart gird, balanced loads for the connectors, as well as enhanced self benefit for EV

customers. This paper aims to propose a new idea for charging back-end management system, which maximizes the benefits of EV owners through low-cost charging.

EV owners are incentivized to charge during off-peak hours (when electricity prices are low), and it is the CSMS performs staggered charging spontaneously rather than human supervision. With the above measures, this back-end management system will be beneficial to maintain grid balance and improve power utilization.

1.4 Research Methodology

This general guidance methodology follows Minimum Viable Product (MVP) method. The term, MVP was first coined by Frank Robinson in 2001. In the Ries definition, the product development process can be reduced by combining business-driven hypothesis experimentation and iterative product releases. Building a product iteratively based on the needs of early customers could lead to reduced market risks such as expensive product launches and failures [4]. Following the guide of MVP, the specific software development model method, using Scrum [13] and Throwaway prototyping [11], is explained in detail in Section 3.1.

1.5 Delimitations

- The effects of large-scale charging on the grid, especially in terms of overloading the grid and voltage reductions, are not taken into consideration.
- Disregard extreme weather conditions.
- One charging station may have more than two connectors, when more than one work at the same time, the output power of each is less than the maximum value (125kW). When the smart charging algorithm in this paper estimates the expected charging time, the power at the beginning is prevail. The dramatic drop in output power caused by the subsequent connection of other vehicles is not discussed.
- The instant the vehicle is connected to the grid or the charging station terminate one transaction can only be coincident with or delayed from to the instant when CSMS send *remoteStartTransaction* and *remoteStopTransaction* request to the charging facility. The interval is kindly ignored in this paper.

1.6 Outline

This report contains the following chapters:

Chapter 1. Introduction

The first chapter elaborates the problems addressed by this project and its contribution to sustainable development.

Chapter 2. Theoretical Background

The second chapter includes the most popular and advanced research results and application scenarios of smart charging nowadays.

Chapter 3. Methods

The third chapter consists of relevant theory in OCPP and the methodology used during this project along with scientific support for which methods were used and how they were implemented.

Chapter 4. Results

The results chapter collects the results from the methods mentioned in chapter four.

Chapter 5. Discussion

The discussion chapter gives a discussion regarding how the project was conducted along with suggestions for improvements and further work.

Chapter 6. Conclusions

The final chapter presents the findings of this project, the strengths and deficiencies of this project.

Chapter 2

Theoretical Background

2.1 What is OCPP 1.6?

The OCPP is an open-source communication protocol for networked electric vehicle chargers [7]. The vision of OCPP is to make any EV charger work with any charger management software, even if the charger manufacturer and software developer have never met and are building their products on the opposite side of the world.

The OCPP 1.6 proposed by the Open Charging Alliance (OCA) has been applied to more than 40,000 charging facilities in 49 countries, it has become a global standard. This paper studied the OCPP 1.6 and simulated the transmission of the message specified by the protocol.

2.2 What is Smart Charging?

Smart charging in general is a wide-ranging topic that includes several objectives. It can mean that the grid capacity is used in such a manner that consumers are able to charge their batteries fully at any time, even if large groups of consumers wish to fill up simultaneously. Smart can also mean that energy prices can be taken into consideration when charging. Or again smart can be taken as using a local supply of sustainable energy from solar panels. And it is even 'smarter' when the EV driver wishes to be part of the solution.

Within OCPP, smart charging means that a CSMS or Charging Station (CS) gains the ability to influence the (de-)charging power or current of a specific EV, or the

total allowed energy consumption on an entire charging station / a group of charging stations. Different setups can be used.

There may be three different use cases for smart charging in OCPP 1.6.

- Load balancing
- Central smart charging
- Local smart charging

There are more complex use cases possible in which two or more of the above use cases are combined into one more complex system [7]. Both internal load balancing and local smart charging concerns the logic and charging limits within the CS, not CSMS. Considering that the optimization of CS is not the primary option of Scania®, this paper is more concerned with exploring the possible benefits of the second approach, central smart charging.

2.3 What affects hourly electricity price?

Electricity prices generally reflect the cost to build, finance, maintain, and operate power plants and the electricity grid.

- **Production cost**: There are cheaper energies, such as nuclear and hydroelectric power, and more expensive ones, such as thermal power and gas, which hugely depend on fluctuations in the price of raw materials. Fuel prices, especially for natural gas and petroleum fuels, may increase during periods of high electricity demand and when there are fuel supply constraints or disruptions.
- **Weather forecasts**: Extreme temperatures can increase demand for heating and cooling, and the resulting increases in electricity demand can push up fuel and electricity prices. On the other hand, rain and snow provide water for low-cost hydroelectric power generation, and wind can provide low-cost electricity generation when wind speeds are favorable.
- **The bid prices**: The day-ahead price for each hour the following day is established through the process that generating companies send in their bids and offers for the volume they want to purchase or sell [5].
- **Demand**: Demand for heating, cooling, light, and processes varies in response

to demand in terms of economic, technological, and efficiency measures. This is a key contributing factor to price fluctuations, which can occur on an hourly basis.

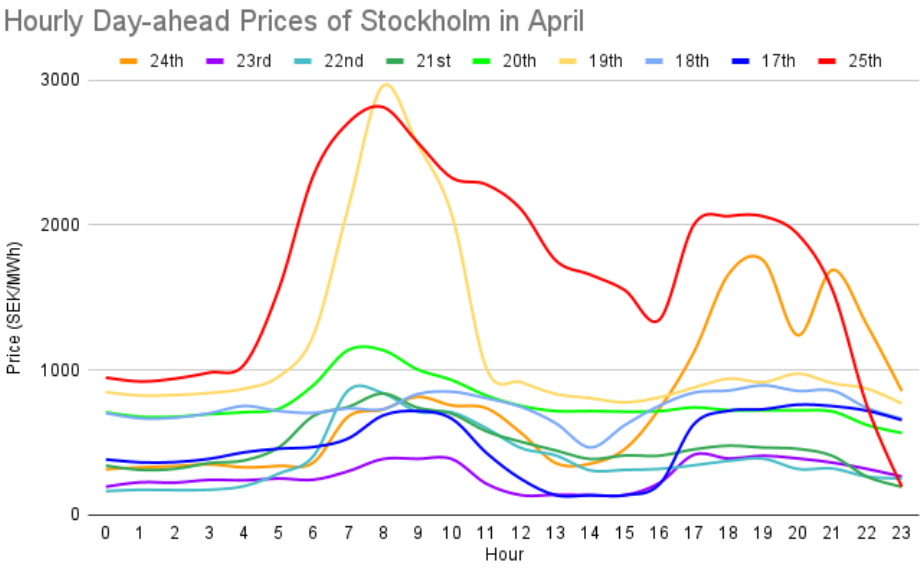

Figure 2.3.1: Hourly electricity price fluctuations at Stockholm in April

Figure 2.3.1 explicitly illustrates the regular fluctuations in the demand market for electricity. Combined with daily experience, the price of electricity reaches its highest point during 8:00-9:00, while valley values usually occur during 14:00-15:00 and 22:00-01:00(day plus one).

2.4 What affects battery capacity estimation?

Accurate estimation of SoC can not only protect the battery, prevent overcharge or discharge, and improve the battery life, but also let the application make rationally control strategies to achieve the purpose of saving energy. In practice SoC does not change linearly with several quantities. [2].

- **Nonlinear characteristics of lithium batteries**The lithium battery system is highly nonlinear, with multi-spatial scale (such as nanometer active materials, millimeter cell, and meter battery pack, etc.) and multi-time scale aging, making it difficult to accurately track the trend.
- **Battery internal properties**: Usable capacity, charge/discharge rate, temperature, cell age, self-discharge etc. cannot be obtained by direct

measurement approach and is easily affected by environment.

- **Energy loss in the process**: Not all energy that is supplied by the charging station goes into the battery. Some is consumed by computers, fans, pumps, heaters etc.

2.5 Related Work

2.5.1 Battery capacity estimation

Battery capacity estimation algorithms can be roughly divided into three categories: direct measurement method not based on battery model, the black box battery model, and the ones based on the state space battery model [19].

The first one craves close attention to the battery's voltage, current, internal resistance, impedance and other correlated battery parameter variables. This method requires extremely high sensitivity of the sensor and its focus is so narrow that turns a blind eye to the truck as a whole. The battery modelling is in need for the last group, while this project neither has nor needs related prior knowledge.

That being so, this paper prefers the second class, the black box battery model. The black box battery model regards the battery as an unknown system, takes the online measurable battery measurements, as the input and the battery capacity as the output. It trains input and output data through some Artificial Intelligence (AI) algorithms, and establishes the relationship between input and output [19], as shown in Figure 2.5.1.

The Artificial Neural Network (ANN) models have good adaptability to nonlinear systems. Sassi et al. conducted a comparative study of ANN and Kalman Filter (KF) for on-board SoC estimation. It turns out that the results of ANN are greatly affected by the size of the sample and the training method, which reduces the practical application ability of this method.

Correspondingly, Xie et al. set their sight on the joint Unscented Kalman Filter (UKF) and Support Vector Machine (SVM) algorithm. According to authors, the experimental results show that the tracking error is less than 1.00%. The new algorithm, SVM and UKF estimation algorithm, achieves higher estimation accuracy for the battery SoC, but also increases the computational complexity.

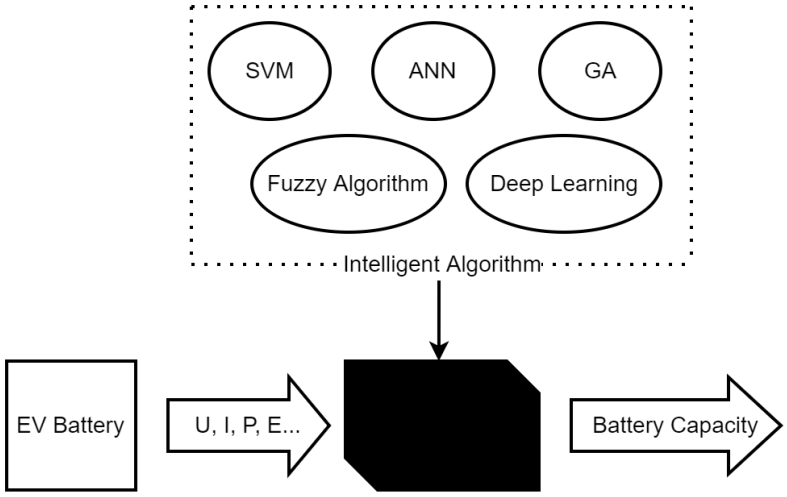

Figure 2.5.1: Battery capacity estimation method based on black box battery model.

There is a trade-off between high-precision estimation and real-time performance on every occasion. The goal of this paper is to be application-oriented, the proposed battery capacity estimation algorithm should prioritize cutting down the time consumption, which to some extent means abandoning the keen pursuit of perfect accuracy.

2.5.2 EV smart charging

When it comes to EV smart charging, related work targets single or multiple aspects such as *Vehicle-to-charging station assignment*, *offline* or *day-ahead planning* and *real-time planning* [1].

The *vehicle-to-charging station assignment* problem deals with assigning EVs to charging stations to optimize infrastructure usage or minimize distance driven, which are not relevant to the purpose of this paper.The *offline* or *day-ahead planning* approaches schedule EV charging well in advance and do not reckon with messages from the charging stations. For that reason, this paper prefers *real-time planning* approaches that dynamically create EV schedules while taking different objectives into account [1].

In the field of *real-time planning*, Nour et al. used a fuzzy logic controller to control and manage the EV charging process to maximize electric utility and EV owner benefits. According to the authors, the experiment results proved this method reduced the impacts of EVs charging on the distribution network. It inspired the model in this

paper, where CSMS played the role of decision maker by sending a schedule with *ChargingProfile* or *RemoteStopTransaction* command.

Yu, Chen, and Tong formulated the large-scale EV charging problem as a stochastic dynamic programming and brought forward Threshold admission and greedy scheduling (TAGS) to maximize operation profit. Admittedly, the scope of this paper is limited to single-charge demands and does not consider a mix of storage, local renewable energy sources, the idea of greedy scheduling still sets the tone for this paper.

2.5.3 Peer Research

The Mobility House

Smart charging for The Mobility House®means making use of load management, i.e. the efficient charging of several electric cars at one location within a given connected load.

Prio.	Charge Point	Status	Power
☆	TMH_000137	CHARGING	3.2 kW
☆	TMH_000163	CHARGING	3.4 kW
☆	TMH_000141	AVAILABLE	-

(a) Before prioritising TMH_000137.

Prio.	Charge Point	Status	Power
★	TMH_000137	CHARGING	3.5 kW
☆	TMH_000163	CHARGING	3.7 kW
☆	TMH_000141	AVAILABLE	-

(b) After prioritising TMH_000137.

When the charging speed of one Charging Point (CP) is too slow, a significant increase in output power can be achieved by adjusting the priority of the current CP. As shown in the figure 2.5.2b, a rapid jump from 3.2kw to 3.5kw for CP TMH_000137 can be seen in a second.

Kempower

Kempower®gives drivers' permission to slow charging mode, via booking a connector in advance. It is undeniable that fast charging mode is to charge the battery quickly with high current, which may cause damage to the battery and shorten the battery life. On the contrary, slow charging has relatively low charging current and power, which is better for the battery life.

Kempower achieved this goal with **Smart Charging OCPP profile**, which is fully supported in the full scope as specified in OCPP standard.

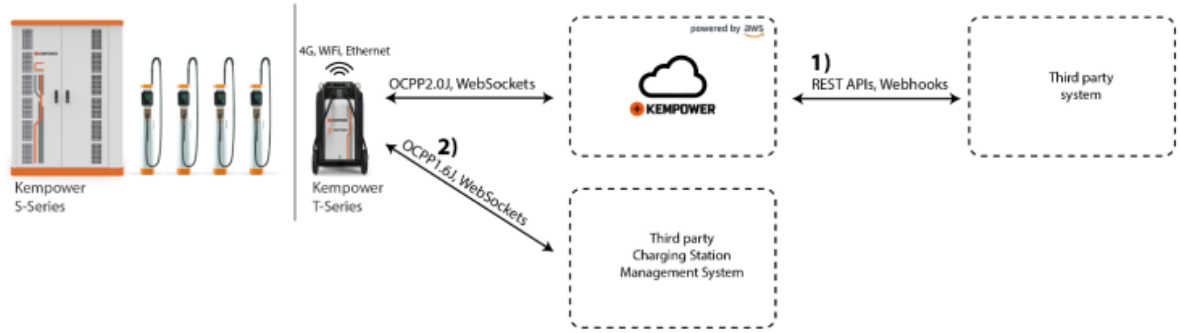

Figure 2.5.3: Kempower provides the direct OCPP 1.6J connectivity to the charger for use.

CarMediaLab

CarMediaLab®is highly compatible with the experimental setting of this paper, both follow OCPP 1.6/2.0.1 and ISO 15118.

Their Smart Charging solution invloes a basis for the innovative concept, Vehicle to Grid (V2G), feeding energy back into the grid. Nevertheless, they have long charging times, usually require new infrastructure, restricted ranges, and possess unfavorable battery effects.

Chapter 3

Methods

3.1 Research methodology

3.1.1 Scrum

This project is expected to start from 17 January to 30 May, approximately 20 weeks. Making allowance for the time constraint, agile software development is the evident choice for this paper.

Scrum is probably the most popular agile model. Scrum iterations, Sprints, are brief, time-boxed periods, usually 2-4 weeks long and they are preceded with thorough planning and previous sprint assessment [13]. The Sprint in this article strictly follows the following steps.

1. Planning
2. Design
3. Implementation
4. Unit Testing
5. Working increment Deployment
6. Evaluation

Overall there are four Sprints in this paper: The first Sprint empowers the cloud CSMS listen to the OCPP messages from the charging station; The second one plans to implement remote switches with charging stations; Then the following Sprint

estimates the battery capacity and the time needed to complete the *targetSoC*; And last but not least, the final Sprint reads the hourly price from Nord Pool Application Programming Interface (API), and deliver a practical intelligent charging solution.

3.1.2 Throwaway prototyping

As clarified in Section 1.3, given the hassles of booking a real truck and the limitations of Amazon free tier account, the most efficient way is to build a local virtual client, which plays the role of CP and ensures the two-way communication is error-free before deploying the latest CSMS to the cloud and testing.

Throwaway prototyping is a model created with the intention of discarding it after the testing process [11]. Throwaway prototypes are quick to create. And because they will be discarded, there is no need to further refine the model, which dramatically cuts down on development time and costs.

There are two basic requirements for a local virtual client.

- Includes the ability to build interfaces.
- Build to visualize the results of user requirements.

During the development of CSMS, the local client terminal output is of great help in checking our expectations and fixing implicit bugs. This prototyping method is best used to provide proof of concept. It does not show much beyond the basics.

3.2 Main Framework

As shown in figure3.2.1, here are four main characters in this paper: **CSMS**, **CS** (also seen as CP and Electric Vehicle Supply Equipment (EVSE)), **EV**, and **Clients** (both EV drivers and CSMS developers).

WebSocket and HTTP

WebSocket can be used if there is a need for any real-time updates or continuous data streams are transmitted over the network. While **HTTP** is recommended when old data is not required very frequently or fetched only once can be queried by the simple HTTP request.

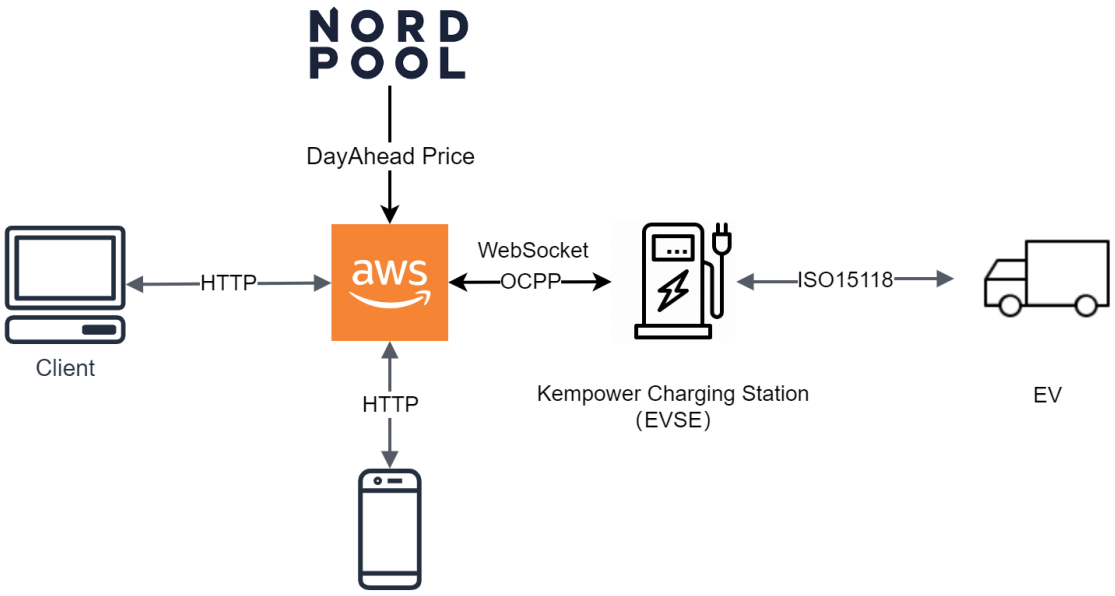

Figure 3.2.1: The main framework of the back-end charging management system.

In this project, as long as the charging is in progress, charging stations keep sending periodic signals, *MeterValues.req* and *StatusNotification.req*, to CSMS. In this scenario, it is better for EVSE and CSMS to follow WebSocket instead of HTTP.

3.2.1 Cloud Deployment

The web server runs on an Amazon Elastic Compute Cloud (EC2) instance using Amazon Linux, and the MySQL database is an MySQL Database (DB) instance in Amazon Relational Database Service (RDS). Both the Amazon EC2 instance and the DB instance run in a Virtual Private Cloud (VPC) based on the Amazon VPC service.

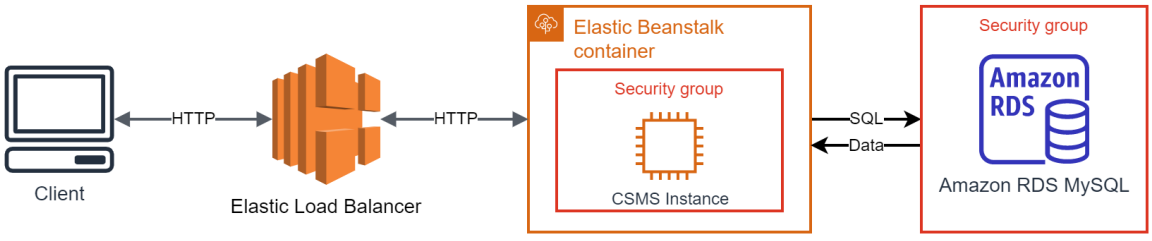

Figure 3.2.2: The cloud framework of server Amazon Web Service (AWS) deployment.

Amazon Elastic Beanstalk

Amazon Elastic Beanstalk is chosen as the cloud deployment platform in this paper, for it is a highly spoken and easy-to-use service for deploying and scaling web applications. By simply uploading our code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring.

3.3 Entities

This paper identified **five** essential entities, and each corresponds to one table in MySQL DB instance. Figure 3.3.1 is the Unified Modeling Language (UML) class diagram explain the relationship between entities.

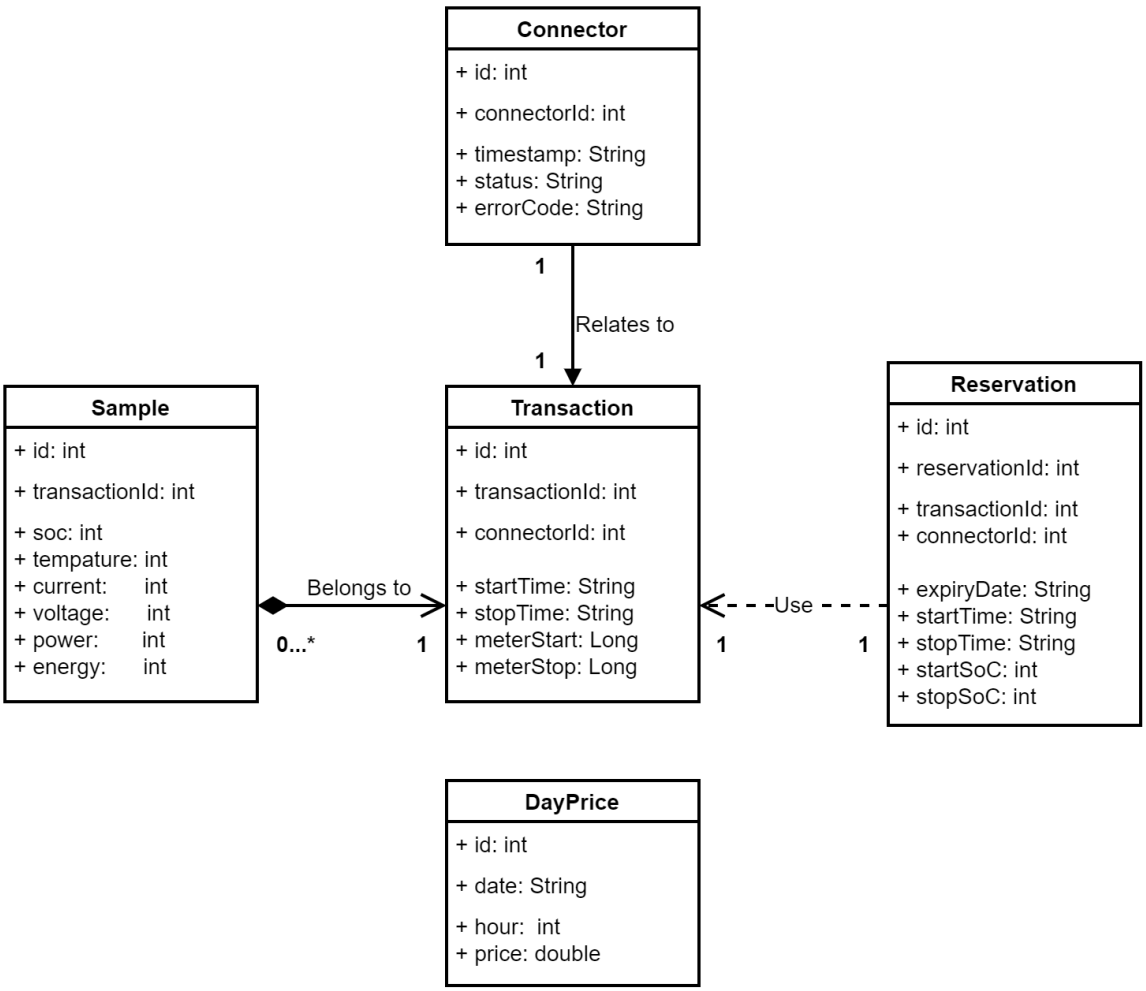

Figure 3.3.1: UML Class Diagram for 5 entities in MySQL DB instance.

3.3.1 Transaction

RemoteStartTransactin and RemoteStopTransaction

1. CSMS can request a charge point to start a transaction by sending a *RemoteStartTransaction* request.
2. Upon receipt, if the CP is capable to start a transaction (the status of its connector is *Available*), it will send a *StartTransaction* request back to CSMS.
3. If CSMS accpets the *StartTransaction* reuqest, it will assign a random *transactionId* and attach it to the confirmation response.
4. After receiving confirmation response, CP starts the transaction in the same way as described in *StartTransaction*.
5. During the transaction, CP keep updating CSMS with the charging process via *MeterValues* (including sample values like timestamp, SoC, temperature, etc.) and *StatusNotification* (including the latest heartbeat, status and error code of connectors).

In the general run of things, transactions are terminated in two ways:

- The driver manually presses stop button and plug off the charger.
- CSMS can request a Charge Point to stop a transaction by sending a *RemoteStopTransaction.req* with the identifier of the transaction. In like manner, CP shall reply with *RemoteStopTransaction.conf* and a status indicating whether it has accepted the request and a transaction with the given transactionId is ongoing and will be stopped. If yes, CP will send *StopTransaction* request to CSMS and the whole transaction finally stops when CP receives the *StopTransaction* confirmation from CSMS.

It is worth noting that the *connectorId* and *transactionId* of the same transaction are consistent (See Figure3.3.2).

3.3.2 Sample

MeterValues

CP may sample the electrical meter or other sensors or transducer hardware to provide extra information about its meter values. Each *MeterValue* element contains

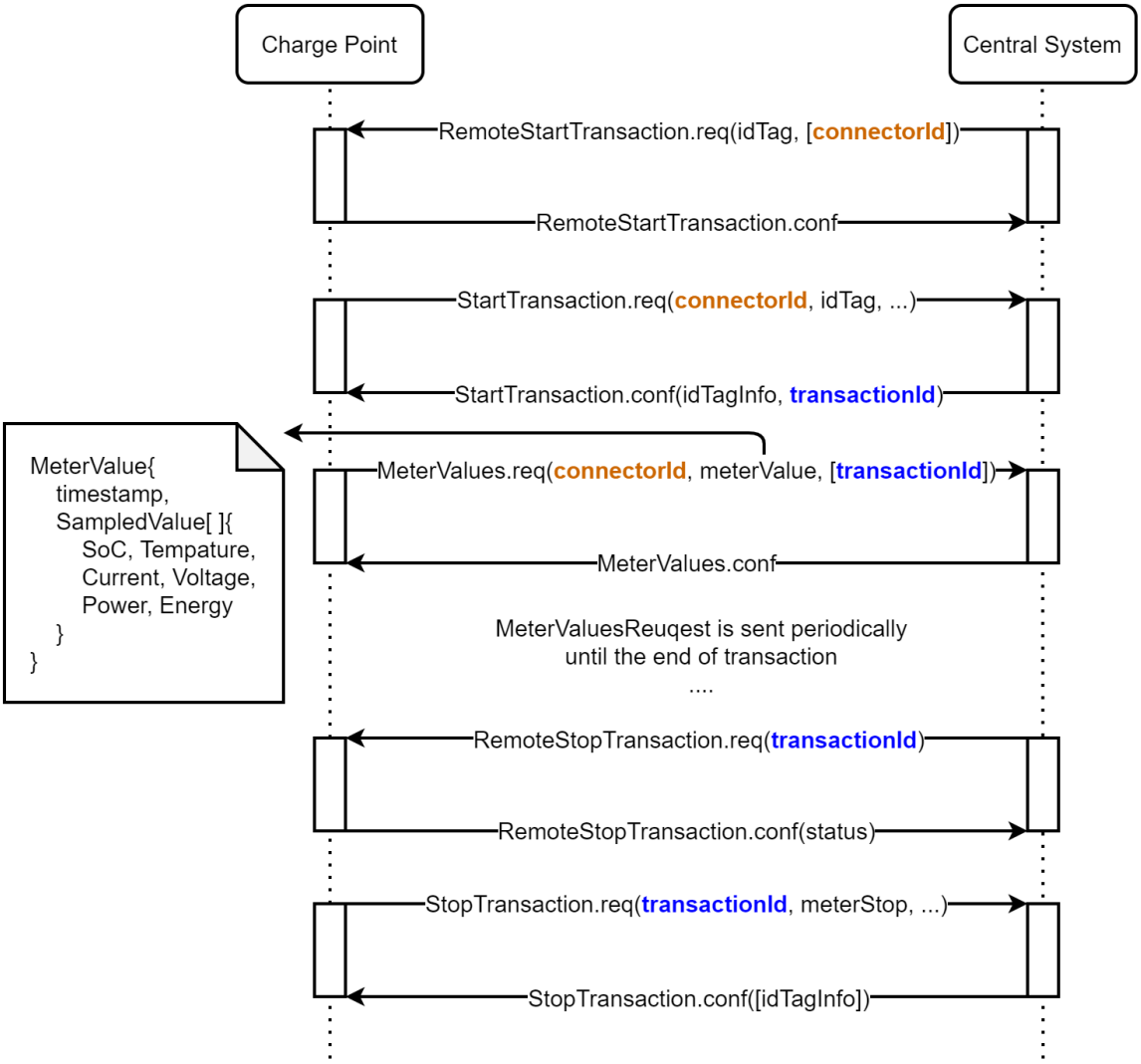

Figure 3.3.2: The sequence diagram of Remote Start and Stop Transaction

a timestamp and a set of one or more individual *SampledValue* elements, all captured at the same point in time [7]. Measurements included in *MeterValues* from Kempower®Charger are listed in table 3.3.1.

3.3.3 Reservation

Comparing to uncontrolled charging, the reservation booking prescribes the client to set up *targetSoC* and *departure time*, and armed with this information, some of the barriers to realizing a smart charging solution have been removed.

Table 3.3.1: Measurements included in *MeterValues* from KemPower Charger.

Measurement	**Notation**
SoC	EV Battery State of Charge %, if known
Temperature	Temperature (Celsius) inside the Charger or Satellite housing
Current.Import	Charging current from Charger to EV (amps)
Voltage	Charging voltage (volts)
Power.Active.Import	Charging power (watts)
Energy.Active.Import.Regsiter	Energy outputted from Charger to EV (watt-hours)

3.4 Smart Charging

3.4.1 Energy and time estimation

OCPP1.6 does not send the requested energy as an absolute value, while it allows sending SoC that represents the current state of the vehicle battery as a percentage. For example, $SoC = 80\%$ means that 20% of the battery needs to be charged with energy. Hence, the battery capacity of the current EV is needed before calculating the expected energy and time. In the experiment, measurements *SoC* and *Energy.Active.Import.Regsiter* included in *MeterValues* (See Table 3.3.1) are used to estimate the expected energy.

Moving Averages

Moving Average (MA) is a calulation to analyze data points by creating a series of averages of different subsets of the full data set. The two most popular types of moving averages are the Simple Moving Average (SMA) and Exponential Moving Average (EMA). EMA will probably experience more short-term changes than a correspnonding SMA, while we need to reduce the influence of extreme values on the final result. In this case, we use SMA to estimate the total battery capacity.

$$C = \frac{\sum_{i=0}^{N-n} \frac{E_{i+n} - E_i}{SoC_{i+n} - SoC_i}}{N - n} \tag{3.1}$$

Equation 3.1 represents the battery capacity estimation process, where C represents the estimated battery capacity; N represents the size of samples; n represents the

moving window size minus one, and they both have a significant effect on the estimated battery capacity:

- When collecting the sample data, the charging point keeps charging EV. If N is too large, the battery may be fully charged before the smart-charging algorithm is implemented.
- If N is too small, the accuracy of results will be insufficient.
- $n < N$
- If n is too small, the adjacent SoC values might be the same while the result will be $Infinity$.

In this experiment, we set $N = 5$ and $n = 2$. With the estimated battery capacity, the energy and time needed to complete the charging transaction are calculated as equation 3.2 and 3.3.

$$E = \frac{C}{SoC_{target} - SoC_{start}} \tag{3.2}$$

$$T = \frac{E}{mid(Power)} \tag{3.3}$$

In which, E represents the expected energy; T represents the expected time; $Power$ represents the *Power.Active.Import* from *MeterValues*. Furthermore, the output charging power in equation 3.3 is the median of the samples (the *Power.Active.Import* of the third sample). The output power of the first sample is not stable enough, because the charging station is still heating up and starting. That is why CSMS waits until the third clock cycle, waiting for the output power to stabilize.

3.4.2 Greedy Algorithm

Single-Interval Scheduling Maximization

This smart charging problem here is formulated into the single interval scheduling maximization. The hourly electricity price is the weight; the interval is an hour; every hour between the start time and departure time is one candidate; how to charging the EV with the lowest price equals to how to schedule the shortest path to the final goal, *targetSoC*.

The following Greedy Algorithm (GA), called Earliest deadline first scheduling, does find the optimal solution for single-interval scheduling.

1. Select the interval x, with the lowest price.
2. Remove x from the candidate intervals.
3. Repeat until the *targetSoC* is reached or the set of candidate intervals is empty.

Why choose GA?

If a greedy algorithm can be proven to yield the global optimum for a given problem class, it typically becomes the method of choice because it is faster than other optimization methods like dynamic programming. In this context, the benchmark is single and simple, which is also called as **greedy choice property**.

Technically, the greedy choice property here is only **cost**. Shorter waiting time certainly shares equal importance when it comes to actual production, while here charging is guaranteed to be done within the budget.

As shown in figure 3.4.1, for uncontrolled charging, the *dayPriceService* only fetches the price list in 2 hours (current hour and next hour). It has to mention that quick-charging of Scania hybrid electric trucks, which supports charging to 80% capacity in just 35 minutes.

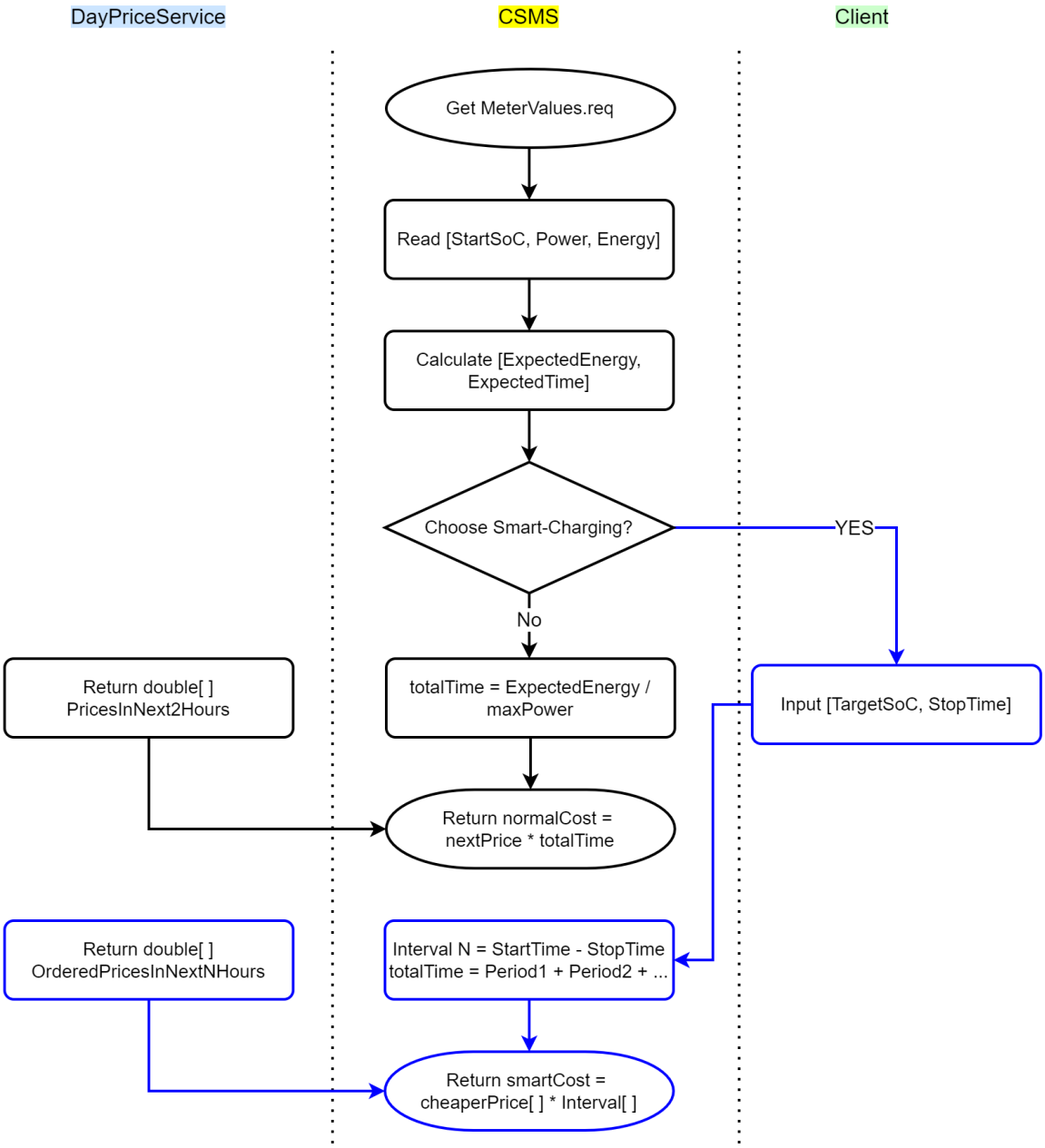

Figure 3.4.1: Smart charging algorithm combined with hourly electricity price.

Chapter 4

Experimental Setup

4.1 Experimental environments

Table 4.1.1: Software/Tools used for application development.

Role	Software	Brief
Back-end Development Framework	Spring Boot	Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications.
Programming Language	Java 11	Java is the most commonly used programming language of the current back-end development.
Web Engine	Thymeleaf	Thymeleaf is a modern server-side Java template engine for both web and standalone environments.
Integrated Development Environment (IDE)	Intellij IDEA	Intellij IDEA provides intelligent coding assistance and productive debugging tool.
Web Deployment	Amazon Elastic Beanstalk	Amazon Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java.
Open-source Database	Amazon RDS MySQL	Amazon RDS provides easy setup, operation, and scaling services for a relational database in the cloud.

4.2 Scenario

EVs

Heavy-duty trucks with large battery capacity are the prioritizing objects in this paper. To verify the robustness of the algorithm, a diverse set of trucks with different characteristics is selected to participate in the test (See Table 4.2.1).

Table 4.2.1: List of trucks attended to the test.

Model	Battery capacity(kWh)
Scania Citywide BEV	254 - 330
Scania BEV	250 - 300
Scania PHEV	30
Volkswagen ID.4	77
Volvo FE Bus	200 - 265

*All data from public content on the web

EVSEs

Kempower®, as one of the Scania potential partner manufacturers, possesses several comprehensive features, such as OCPP 1.6/2.0, cloud-based back-end, service, and management dashboard.

The EVSEs in this project are Kempower C801 P160 N®charging stations. The specific technical parameters are given in Table 4.2.2.

Table 4.2.2: Technical data of Kempower C801 P160 N[3]

Technical indicator	Standard value	Unit
Output power	40 - 125	kW
Output current	250 - 300	A
Output voltage	200 - 800	Volts Direct Current (V_{DC})
Efficiency	95	%
Operating temperature	-30 - +40	C°

Smart pricing

It is not possible to minimize energy costs under fixed tariffs, in which the electricity price remains constant throughout the day. The potential for cost minimization with smart charging exists only under time-varying tariffs, in which the price of electricity depends on the time of consumption [14]. Electricity consumption costs in this paper are determined by the electricity prices from Nord Pool and the amount of energy consumed.

Time of planning

There are two smart charging plannings in this paper: day-ahead (or offline) planning and real-time (or online) planning. Evaluating the results of day-ahead planning in isolation includes arrival time, departure time and the SoC. Real-time planning requires decisions to be taken as soon as possible on each EV arrival as well as departure since premature departures can create new flexibility [1].

Reservation

Drivers may miss or start reservations early for various reasons, and unused reservations is a waste of resources. For that reason, in the reservation form, CSMS automatically extracted the start charging time and the start SoC from the first *MeterValues* arrived with the expected *transactionId* (See Figure 5.1.3).

Chapter 5

Result Analysis

5.1 Implementation

It is worth noting that every Kempower®DC fast charger is connected to the Internet and the Kempower Cloud Service (via the included 4G modem, or Ethernet/WiFi if set up), and support a dual OCPP mode. This brings great convenience to our experimental development, in place of caring about the hardware implementation of the charging station, but only about the construction of the cloud management system.

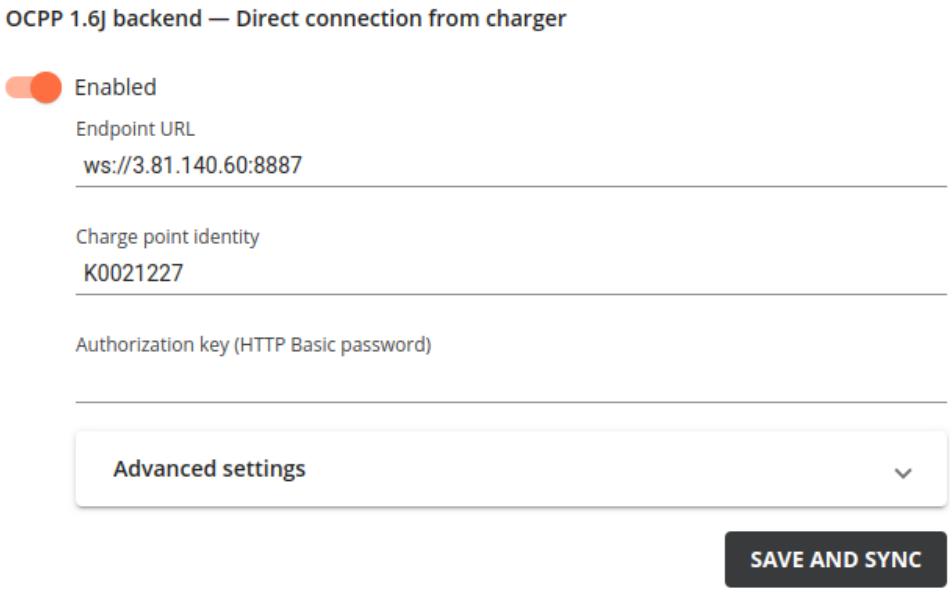

Figure 5.1.1: Direct OCPP 1.6J connection with Kempower charger.

5.1.1 Transaction

Figure 5.1.2 is a screenshot of Kempower®back-end dashboard, where the transaction 15205 was successfully started and stopped remotely by our server terminal.

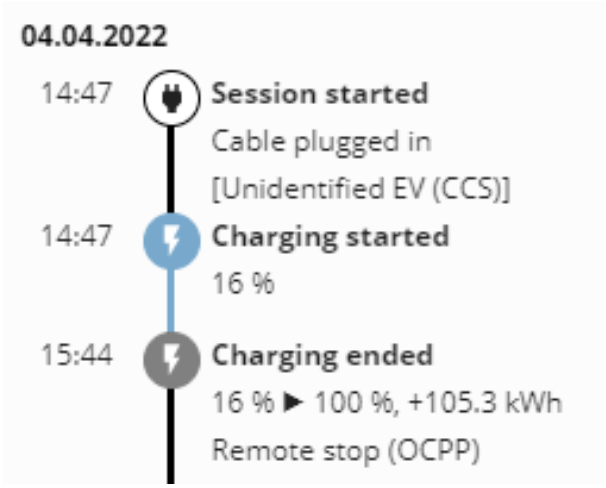

Figure 5.1.2: Remotely start and stop transaction 15205.

5.1.2 Reservation

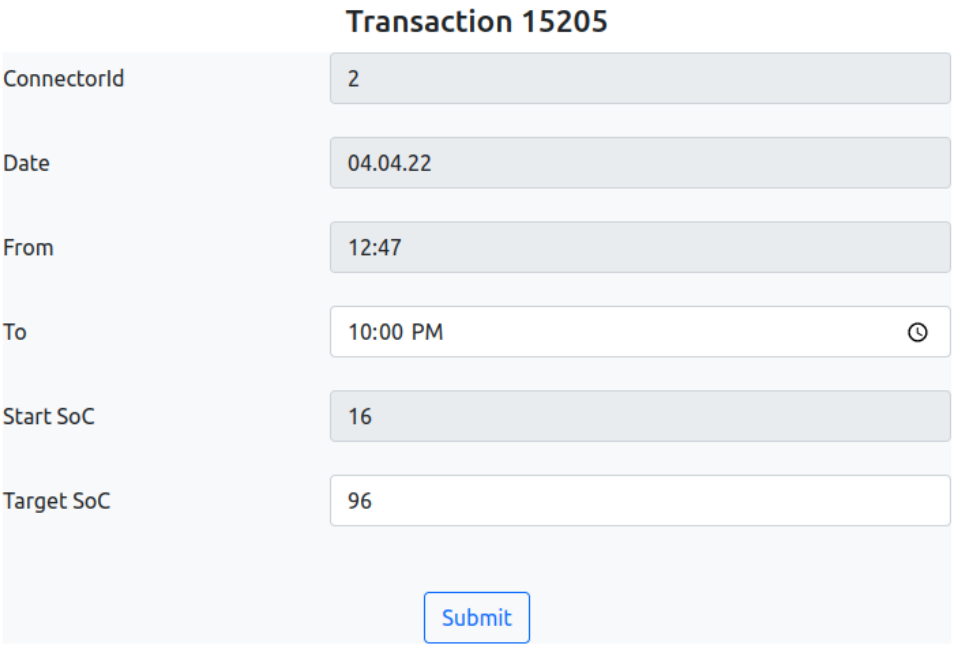

Figure 5.1.3: CSMS: Reservation form for transaction 15205.

Figure 5.1.4 illustrates the day-ahead smart charging planning for transaction 15205. Contrary to uncontrolled charging, smart charging predicts there a valley value came in two hours and waited until 14:00.

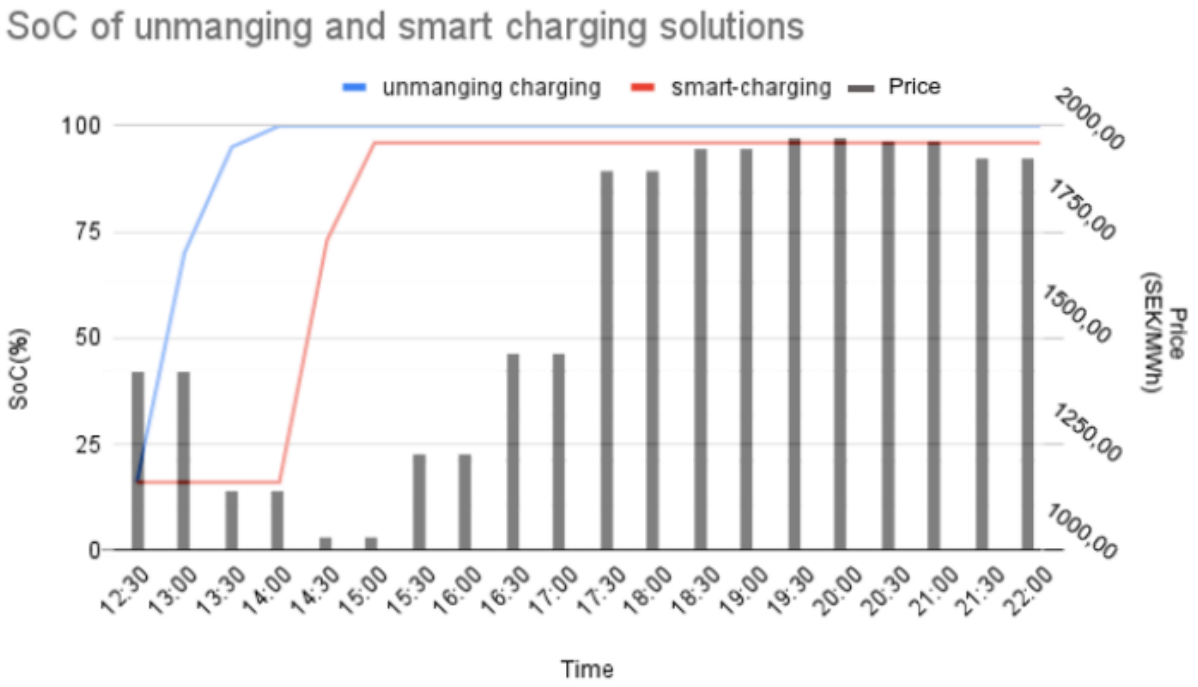

Figure 5.1.4: Uncontrolled and smart charging solutions under the influence of price.

5.1.3 Front-end and back-end integration

The original proposal of this project was divided into two parts, the back-end (CSMS) and the front-end (Mobile Phone Android Application), which are respectively led by the author and another thesis worker, Christine Arkbo.

It makes sense that the back-end takes the responsibility for developing and delivering the API to the front-end, with database access rights and reasonable remote operations of charging stations. Up to the completion of this paper, features that have been successfully integrated so far include heartbeat check and remote switches.

To let the CSMS know that a Charge Point is still connected, a Charge Point sends a heartbeat after a configurable time interval (See Table 5.1.1). Upon receipt of a *Heartbeat.req* Professional Development Units (PDU), the CSMS will respond with a *Heartbeat.conf*. The response JSON file contain the current time of the Central System, which is recommended to be used by the Charge Point to synchronize its internal clock [7].

> With JSON over WebSocket, sending heartbeats is not mandatory. However, for time synchronization it is advised to at least send one heartbeat per 24 hour. [7]

All of the following interfaces have a common beginning of the Uniform Resource

Identifier (URI), `http://3.81.140.60/api/`, where `3.81.140.60` is the given public IPv4 address from CSMS deployed on Amazon Elastic Beanstalk.

Table 5.1.1: API for heartbeat check.

Path	`heartbeat/{connectorId}`	
Content	Send heartbeat to CSMS.	
Data Type	`id`	int
	`connectorId`	int
	`timeStamp`	ZonedDateTime
	`status`	String
	`errorCode`	String

When CSMS receives the start or stop request to charging transaction from the mobile terminal (See Table 5.1.2), it first checks whether the current session is null, that is, whether there is a EV charging at the listening CP. If so, when the CSMS will open a new dialogue with current Charge Point, as described in Section 3.3.1.

Table 5.1.2: API for remote switches.

$Path_1$	`start`	
$Function_1$	Remotely start a new charging transaction.	
$Path_2$	`stop`	
$Function_2$	Stop the current transaction remotely.	
Data Type	`id`	int
	`transactionId`	int (1xxxx)
	`connectorId`	int
	`startTime`	ZonedDateTime
	`stopTime`	ZonedDateTime

5.2 Prediction precision

5.2.1 Mean Absolute Percentage Error

Mean Absolute Percentage Error (MAPE) is a measure of how accurate a forecast system is. It measures this accuracy as a percentage, and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values. It is used to measure the accuracy of the estimated time in this paper.

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{5.1}$$

In equation 5.1, M represents mean absolute percentage error; n represents number of transactions participated in this test; A_t represents actual time; F_t represents forecast time. The last column of table 5.2.1 is the Absolute Percentage Error (APE) of every transaction.

The final MAPE is **12.547%**, which is an ideal experimental result. From the observation of sample values from transaction 16122 and 12368, there is a sharp drop down in output power once the SoC exceeds 96%, which caused the prediction error to be larger than expectation.

Table 5.2.1: Prediction accuracy of estimated time.

Transaction ID	**Actual Time(h)**	**Estimated Time(h)**	**APE (%)**
16122	1.351	1.633	20.873
17817	0.233	0.250	7.296
11201	0.543	0.480	11.602
11941	0.182	0.183	0.549
12368	0.525	0.400	23.810
18957	0.226	0.250	10.619
12706	1.323	1.150	13.076

5.2.2 Forecast Performance Evaluation

In conclusion, from attentive observation and study on the Kempower®provided measurements, this paper came out with one easy-implemented and highly-applicable method to infer the required energy and time, Moving Average (MA). It permits a swift and robust forecast within five system cycles (One system cycle sends one *MeterValues* request). From the view of battery protection, when condition permits, it is firmly recommended to set up the *targetSoC* to 80%. That is to say, the MAPE of reservation is generally less than 12.547%.

Adaptive parameters

On the basis of the MA method, there are **three** parameters that can be fine-tuned in the estimation algorithm of this paper.

- N, the size of samples.
- n, the moving window size.
- $Power$, the *Power.Active.Import* from *MeterValues*.

Frankly speaking, the current algorithm goes towards the most straightforward implementation path, where the values of N and n are fixed. To give one example of the limited application scenarios, when the battery capacity is 300kWh and the current output power is 4kW, the SoC only increases by 1.33% after one hour of charging. Obviously, in this case, it is impossible for current MA algorithm to estimate the required energy and charging time within the first five clock cycles.

In response to this issue, one way of thinking is to add the feedback loop. Many AI studies suggest that the accuracy rate of neural network models is significantly higher with at least one feedback loop than those without any. To put it in a simple way, we can dynamically adjust the values of N, n and $Power$ by comparison between actual time and estimated time, within a bigger cycle (15 or 20 system cycles).

> When the estimated time is much smaller than the actual time, this may be due to the fact that the actual output power is smaller than the sample median. Therefore, we re-collect five samples and take the new median as the output energy and re-estimate the time;
>
> When the estimated time is exceedingly larger than the actual time, the reason could be that the estimated battery capacity is too high, which might be because a small n. Owing to a small moving window n, the SoC values at the left and right ends may be the same, and when their difference, 0, is used as the denominator in formula 3.1, the estimated battery capacity C is infinite.

Following this track, section 6.3.3 also brings another quick fix, to build a database of EV information, including MAC address and the initial battery capacity.

5.3 Actual Profit

The following representative charging transactions were selected to test the performance of this algorithm. All transactions in table 5.3.1 are high energy demands and started at random time during the daytime. Meanwhile, to increase the credibility of the results, the selected transactions range from various dates intentionally.

Table 5.3.1: Profits from uncontrolled to smart charging solutions.

Transaction ID	Date	Uncontrolled cost(SEK)	Smart cost(SEK)	Profit (SEK)
15205	04-04	61.555	46.258	15.297
13676	04-14	109.622	31.565	78.057
12882	04-25	174.208	12.606	161.602
16463	04-26	119.772	3.684	116.088
11764	04-26	249.274	7.622	241.652
17186	04-27	208.738	27.696	181.042
12486	04-28	221.446	17.649	203.797

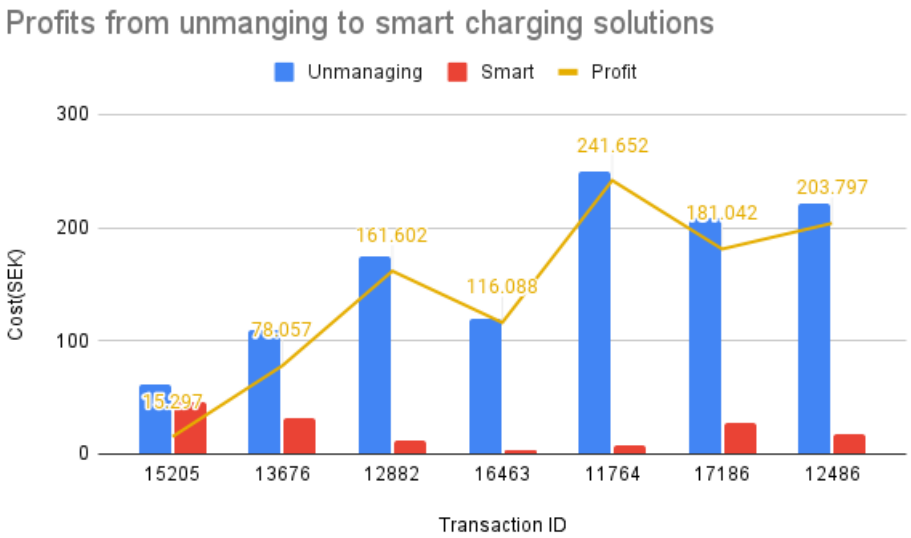

Figure 5.3.1: Profits from uncontrolled to smart charging solutions.

Figure 5.3.1 demonstrates that the proposed smart charging method indeed leads to considerable benefits of customers and the company, compared with uncontrolled charging. Through multiple experiments, the actual profit went above and beyond the call of duty, with such a giant margin on a single charging transaction.

5.4 Validity and Reliability Analysis

The *Actual Time* and *Consumed Energy* for the entire chapter were sourced from the real truck charging data, with no fabrication. Electricity prices are fetched by the API from a third-party website, Nord Pool Market data, all of which is publicly accessible and easy to track the history.

Where there may be a controversy for validity is the profit difference between uncontrolled charging and proposed smart charging solutions. To give a straight intuition of the profit difference, in this section, all departure time are set to 23:59 of that day, which means that it assumed all smart charging solutions started charging at the lowest or second lowest price of the day. As mentioned earlier in section 2.3, the local lows in electricity prices usually come around two o'clock (lunch break) and eleven o'clock in the evening (sleep time), so the assumptions match up to real productive life.

On the other hand, the question to be answered regarding the reliability is, if the current solution, MA along with GA,is the best tool for implementation, compared to peer studies.

It cannot be stressed enough that this paper is under a unique context, in which cost is the one and only greedy choice property, hence the local optimum must be a part of the global optimal solution, Greedy Algorithm (GA) effortlessly became the first choice. The space complexity of GA is always $O(1)$, while the time complexity is controversial, varying from $O(NlogN)$ to $O(N^2)$. Reckoning with the span of charging transactions is 5 to 8 hours in usual, the delay of time complexity has no effect here.

Chapter 6

Conclusions

6.1 Strengths

6.1.1 High commercial value

Most of preceding implementations of OCPP server were simply listening to the requests from virtual clients [17] [9]. However, with the comprehensive hardware support from Scania®, this project has conducted tests in the field and coped with the challenges of real-world charging stations. Furthermore, cloud connection ensures wireless software updates, remote re-booting, easy data management and hassle-free maintenance.

In this way, the struggling to bring this project into production is reduced to a great extent, compared with other contemporaneous approaches.

6.1.2 Avoiding overwhelming data

Consistent effort is made to steer clear of overwhelming data operations like data reading, computing and storage. By way of illustration, there are continuous discharge behaviors for charging station to make up for the power consumed by the battery self-discharge. (See Figure 6.1.1)

During this process, the SoC of battery stays unchanged, $SoC = 100\%$, but the fluctuation was monitored and recorded by CSMS, as listed in figure 6.1.2. These data has no contribution to improving this paper, but it did occupy a significant amount of valuable cloud storage and cloud service resources. Wherefore when a transaction has

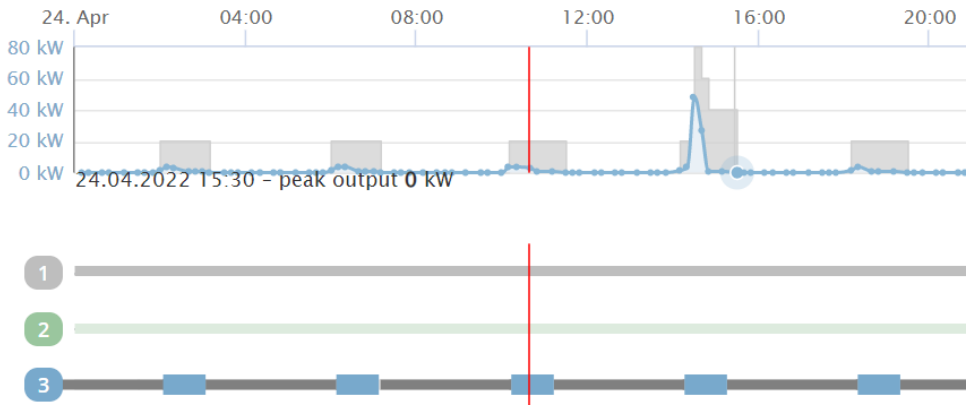

Figure 6.1.1: Charging station continuous discharge behavior.

more than three $SoC = 100\%$ recordings, CSMS will terminate the current session and stop reading the subsequent *MeterValues*.

Samples

SoC (%)	Temperature (Celsius)	Current (A)	Voltage (V)	Power (W)	Energy (Wh)
100	31	3	706	707	0
100	32	78	719	54727	568
100	32	83	724	60009	1580
100	33	84	727	61000	2562
100	33	82	729	59696	3585
100	33	80	731	58400	4538
100	33	78	733	57111	5521
100	33	77	735	56548	6485
100	33	75	737	55275	7396
100	33	76	739	56148	8333

Figure 6.1.2: Charging station continuous discharge behavior.

6.1.3 Bright database management

The price database is not automatically updated every day, instead it may happen when clicking the submit button of the reservation form.

When a reservation form is submitted, *dayPriceService* will check if the current date exists in the price database. If not, it will update itself by deleting all historical data first. And then it saves the electricity price list for today and tomorrow. By plainly maintaining up-to-date data, this project effectively ward off long-term retention of

invalid data in the database. The daily update setting is likely to unknowingly incur exorbitant costs.

On the other hand, bearing the potential value of samples in mind, this server terminal warmly offers the option of copying current snapshots to the local MySQL®DB. This move not only empowers offline training of the black-box battery model but also sets forth a positive precaution against misoperations and database crashes.

6.2 Risks

6.2.1 Authentication

Leonard Bernstein said, to achieve great things, two things are needed; a plan, and not quite enough time. The most urgent need of this project is exploring the achievability of smart charging based on OCPP, and anyone with access to the subordinate URL can remotely control the charging stations or book a reservation, absolutely an unwise risk.

Hence the future development is under a compulsion to add a protective shell, including user authentication and user permission classification, such as guest and administrator.

6.2.2 AWS security group

With rules in figure 6.2.1, any requests follows Transmission Control Protocol (TCP) format can be received by Amazon MySQL DB. The current inbound rule settings are too broad that also expose the server in potential danger.

Security group rule...	IP version	Type	Protocol	Port range
sgr-078c7673dae22ba...	IPv4	All TCP	TCP	0 - 65535
sgr-0d03250745f4781...	IPv6	All TCP	TCP	0 - 65535
sgr-07afda79b428c9b63	–	HTTP	TCP	80

Figure 6.2.1: Inbound rules of AWS security group.

Figure 6.2.2 is a good example, a screenshot from Amazon Elastic Beanstalk logs, where a malicious web bot keeps pestering CSMS with unreadable bad requests.

Taking this as a lesson, before throwing this project into the real production environment, a better inbound and outbound rules settings are of vital importance.

```
*449773 using uninitialized "year" variable while logging request, client: 172.104.159.48, server: , request: "□□□□□□□□ú□□□wš"
*449773 using uninitialized "month" variable while logging request, client: 172.104.159.48, server: , request: "□□□□□□□□ú□□□wš"
*449773 using uninitialized "day" variable while logging request, client: 172.104.159.48, server: , request: "□□□□□□□□ú□□□wš"
*449773 using uninitialized "hour" variable while logging request, client: 172.104.159.48, server: , request: "□□□□□□□□ú□□□wš"
*449792 using uninitialized "year" variable while logging request, client: 145.239.154.82, server: , request: "□□□□i□□□e□□U□§ärandom1random2random3random4□□□/□"
*449792 using uninitialized "month" variable while logging request, client: 145.239.154.82, server: , request: "□□□□i□□□e□□U□§ärandom1random2random3random4□□□/□"
*449792 using uninitialized "day" variable while logging request, client: 145.239.154.82, server: , request: "□□□□i□□□e□□U□§ärandom1random2random3random4□□□/□"
*449792 using uninitialized "hour" variable while logging request, client: 145.239.154.82, server: , request: "□□□□i□□□e□□U□§ärandom1random2random3random4□□□/□"
*451555 using uninitialized "year" variable while logging request, client: 198.235.24.138, server: , request: "□□□□□È□□□Æ□□÷□~□ô□ûýX!NwiX^‘1Z□i3Úc?‘R¼Ñè”Pó□□hÌ□Ì□À/
```

Figure 6.2.2: Malicious script keeps harassing the server.

Make no mistake, it could never be too careful when exposing interior passages to Internet.

6.3 Future Work

6.3.1 From OCPP 1.6 to 2.0.1

Improvements for better handling of large amounts of transactions

The structure and method for reporting transaction is unified in OCPP 2.0. In OCPP 1.x, the reporting of transaction data is split over the messages *StartTransaction*, *StopTransaction*, *MeterValue* and *StatusNotification*. With the market progressing towards more enhanced scheduling, a need is born for more sophisticated handling of transaction data. All the *StartTransaction*, *StopTransaction*, and transaction related *MeterValue* and *StatusNotification* messages are replaced by ’TransactionEvent’. The *StatusNotification* message still exists, but only for non-transaction related status notifications about connector availability[8].

Extended Smart Charging

In OCPP 2.0.1 Smart Charging functionality has been extended (compared to OCPP 1.6) to support:

- Direct Smart Charging inputs from an Energy Management System (EMS) to a Charging Station.
- Improved Smart Charging with a local controller.
- Support for integrated smart charging of the CSMS, EVSE and EV (ISO15118-1).

These features open up more possibilities for future expansion of smart charging capabilities, which is another reason why it is recommended to upgrade OCPP.

6.3.2 From one-to-one to one-to-many

To deliver a working product in the shortest possible time (MVP), the server have not set up a multi-threaded service for the WebSocket connection. ”Come last, Serve first”, to put it in a simple way, if truck A arrives at 8:00 and truck B arrives at 8:20, from 8:20, the current session (See Figure 6.3.1) only exists between CSMS and truck B.

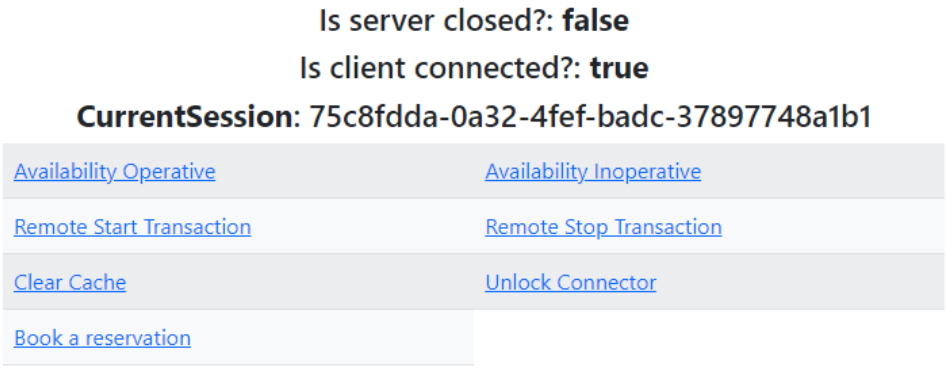

Figure 6.3.1: CSMS: home page.

Combined with the relevant knowledge of multi-threaded communication, it is undoubtedly technically feasible to implement this feature.

6.3.3 Better data utilization

Thanks to the advice from Hampus, there are downsides to estimating the total capacity of the truck. The calculated results as intermediately generated data are not fully utilized, and this process is repeated every time when the charging transaction starts.

Another attainable solution is establishing a database for company trucks in advance. Each is identified by the MAC-address, and the data could be reused to train the estimation model. When that that same vehicle is charging the next time, this would speed up the phase of energy (and time) estimation and thereby improve the overall efficiency of the algorithm.

Bibliography

[1] Frendo, Oliver, Gaertner, Nadine, and Stuckenschmidt, Heiner. “Real-time smart charging based on precomputed schedules”. In: *IEEE Transactions on Smart Grid* 10.6 (2019), pp. 6921–6932.

[2] Hasan, ASM Jahid, Yusuf, Jubair, and Faruque, Rumana Binte. “Performance comparison of machine learning methods with distinct features to estimate battery SOC”. In: *2019 IEEE Green Energy and Smart Systems Conference (IGESSC)*. IEEE. 2019, pp. 1–5.

[3] *Kempower C series General Brochure Lowres*. English. Kemppi Group.

[4] Lenarduzzi, Valentina and Taibi, Davide. “MVP explained: A systematic mapping study on the definitions of minimal viable product”. In: *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2016, pp. 112–119.

[5] Lindström, Markus. *Forecasting day-ahead electricity prices in Sweden: Has the forecasting accuracy decreased?* 2021.

[6] Nour, Morsy, Said, Sayed M, Ali, Abdelfatah, and Farkas, Csaba. “Smart charging of electric vehicles according to electricity price”. In: *2019 international conference on innovative trends in computer engineering (ITCE)*. IEEE. 2019, pp. 432–437.

[7] *Open Charge Point Protocol 1.6*. Open Charge Alliance. 2015.

[8] *Open Charge Point Protocol 2.0.1*. Open Charge Alliance. 2020.

[9] Pruthvi, Thota Venkata, Dutta, Niladri, Bobba, Phaneendra Babu, and Vasudeva, B Sai. “Implementation of OCPP protocol for electric vehicle applications”. In: *E3S Web of Conferences*. Vol. 87. EDP Sciences. 2019, p. 01008.

[10] Ries, Eric. “Minimum viable product: a guide”. In: *Startup lessons learned* 3 (2009), p. 1.

[11] Sadabadi, Ali Tizkar and Tabatabaei, Naser M. “Rapid prototyping for software projects with user interfaces”. In: *no. October* (2009).

[12] Sassi, Hicham Ben, Errahimi, Fatima, Es-Sbai, Najia, and Alaoui, Chakib. “Comparative study of ANN/KF for on-board SOC estimation for vehicular applications”. In: *Journal of Energy Storage* 25 (2019), p. 100822.

[13] Schwaber, Ken and Sutherland, Jeff. “The scrum guide”. In: *Scrum Alliance* 21.1 (2011).

[14] Visakh,
Arjun and Manickavasagam Parvathy, Selvan. “Energy-cost minimization with dynamic smart charging of electric vehicles and the analysis of its impact on distribution-system operation”. In: *Electrical Engineering* (2022), pp. 1–13.

[15] Xie, Fei, Wang, Shunli, Xie, Yanxin, Fernandezb, Carlos, Li, Xiaoxia, and Zou, Chuanyun. “A novel battery state of charge estimation based on the joint unscented kalman filter and support vector machine algorithms.” In: *International journal of electrochemical science* 15.8 (2020).

[16] Yu, Zhe, Chen, Shiyao, and Tong, Lang. “An intelligent energy management system for large-scale charging of electric vehicles”. In: *CSEE Journal of Power and Energy Systems* 2.1 (2016), pp. 47–53.

[17] Zhao, Chaoyue and You, Xiangdong. “Research and Implementation of OCPP 1.6 Protocol”. In: *2017 2nd International Conference on Machinery, Electronics and Control Simulation (MECS 2017)*. Atlantis Press. 2016, pp. 283–291.

[18] Zhong, Jin, Bollen, Math, and Rönnberg, Sarah. “Towards a 100% renewable energy electricity generation system in Sweden”. In: *Renewable Energy* 171 (2021), pp. 812–824.

[19] Zhou, Wenlu, Zheng, Yanping, Pan, Zhengjun, and Lu, Qiang. “Review on the battery model and SOC estimation method”. In: *Processes* 9.9 (2021), p. 1685.

Appendix - Contents

Appendix A

Source code and demo

- Github: Sun-Xiaoying/OCPP-Server
- Web: OCPP Server

Appendix B

Server dashboard

Click on the link, the home page and top navigation bar are introduced. The top navigation can redirect to four different parts. (See Figure B.0.1)

Figure B.0.1: CSMS: Top Navigation Bar.

From right to left, they are **Connectors**, **Transactions**, **Reservations**, and **Day-ahead Prices**. Figure B.0.2 shows the status and error code of current connectors.

Connectors

ConnectorId	Status	ErrorCode	Timestamp
0	Available	NoError	2022-05-03T15:52:13.535Z
1	Available	NoError	2022-05-04T04:20:02.408Z
2	Charging	NoError	2022-05-04T04:15:24.435Z
3	Available	NoError	2022-05-04T04:19:38.800Z

Figure B.0.2: CSMS: Connectors Page.

In accordance with Client and server library of Open Charge-Point Protocol from openchargealliance.org, The enumeration value of the connector status includes *Available, Preparing, Charging, SuspendedEVSE, SuspendedEV, Finishing, Reserved, Unavailable, Faulted*;

The enumeration value of the connector error code includes: *ConnectorLockFailure, EVCommunicationError, GroundFailure, HighTemperature, InternalError, LocalListConflict, NoError, OtherError, OverCurrentFailure, OverVoltage, PowerMeterFailure, PowerSwitchFailure, ReaderFailure, ResetFailure, UnderVoltage, WeakSignal.*

Next, **Transactions** page recorded every valuable transaction and the following services are allowed. (See Figure B.0.3)

- **If Reserve** provides the day-ahead planning service.
- **Delete** provides manual removal of transactions deemed worthless. (See Figure B.0.4)
- Clicking on any ***transactionId***, the page will be redirected to its sample values.

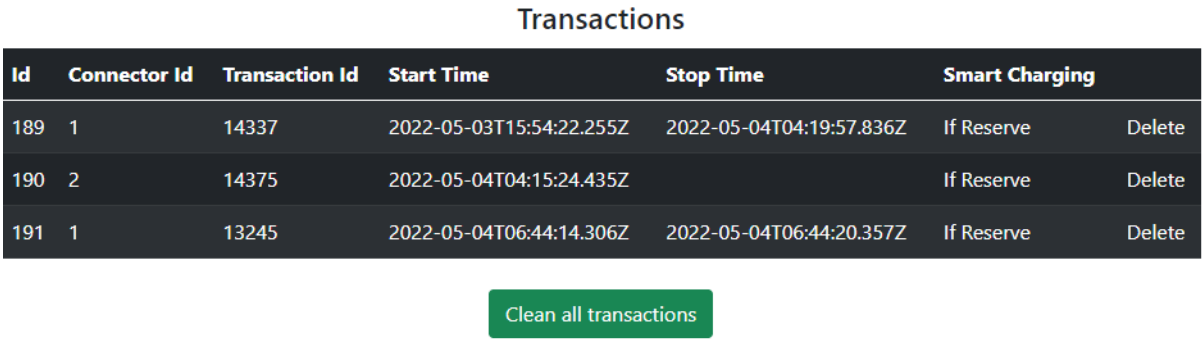

Figure B.0.3: CSMS: Transactions page.

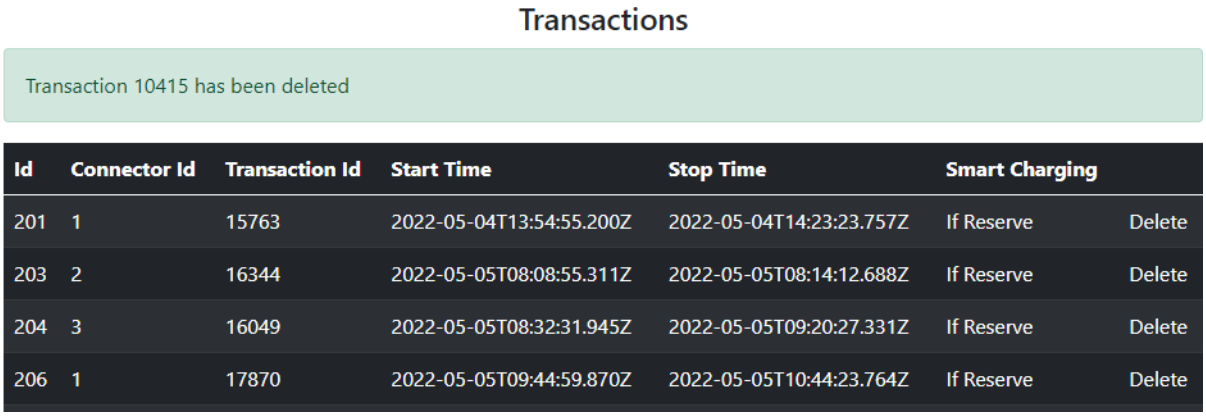

Figure B.0.4: CSMS: delete transaction 10415 page.

Similar to Transactions, **Reservations** page records day-ahead planning and real-time planning made by the clients. (See Figure B.0.5)

Last but not least, **Day-ahead prices** page shows hourly electricity prices and **Refresh prices** can update to the latest date. (See Figure B.0.6)

Reservations

Reservation Id	Transaction Id	Date	Start	Stop	
18953	15763	04.05.22	13:54	19:13	Delete
13875	16319	05.05.22	12:15	21:13	Delete

Figure B.0.5: CSMS: Reservations page.

Nord Pool Dayahead

Prices Updated

Refresh Prices

Date	Hour	Price (SEK/MWh)
06.05.22	0	186.31
06.05.22	1	188.28
06.05.22	2	216.55
06.05.22	3	290.91
06.05.22	4	309.87
06.05.22	5	1314.86
06.05.22	6	2334.35
06.05.22	7	2847.21
06.05.22	8	2723.45
06.05.22	9	2376.4

Figure B.0.6: CSMS: update hourly price list page.

Appendix C

API documentation

Heartbeat

URI format: *http://3.81.140.60/api/heartbeat/{connectorId}*

JSON format:

```
http://3.81.140.60/api/heartbeat/1
{
    "id":2,
    "connectorId":1,
    "timeStamp":"2022-05-16T08:46:42.075Z",
    "status":"Available",
    "errorCode":"NoError"
}
```

Remote start and stop transaction

URI format:

RemoteStartTransaction: *http://3.81.140.60/api/start*

RemoteStopTransaction: *http://3.81.140.60/api/stop*

JSON format:

```
{
    "id":320,
    "transactionId":16330,
```

```
    "connectorId":2,
    "startTime":"2022-05-18T10:00:54.881Z",
    "stopTime":null
}
```

TRITA – XXX-XXX 20XX:XX