Implementing a real-time control algorithm of Triana on SASensor Open Platform

Prof.dr.ir. Gerard J.M. Smit, Dr.ir. Vincent Bakker, Gerwin Hoogsteen, Dr.ir. Omar Mansour, Prof.dr. Johann L. Hurink

Qiang Fu

August 27, 2015

UNIVERSITY OF TWENTE.

Locamation
smart smart grid solutions

Abstract

In terms of electricity generation, Distributed Generation (DG) using Photovoltaics (PV) panels has become more common. To deal with the potential challenges it brought on the existing electricity distribution network, Demand Side Management (DSM) methodologies are developed. Triana, developed by Computer Architecture and Embedded Systems group (CAES) of University of Twente, is one of them. It is divided into three steps. In the first two control steps, it is able to utilize simulation or measurement data off-line to forecast and plan electricity production/ consumption on the device level.

Since prediction and planning errors can be caused by fortuitous behaviors of end users, it is often impossible to foresee it and prevent it from happening. It is however possible to detect these errors and solve them using a real-time control algorithm in the third step of Triana. The design, implementation and testing of this real-time control algorithm are included in this thesis.

In cooperation with Locamation, an Application Programming Interface (API) is added to the Triana to detect prediction and planning errors. Power, voltage and current data of medium voltage (MV) to low voltage (LV) transformers can be gathered in real time from the SASensor open platform. From the SASensor open platform, Triana can access the real-time measurement data (RTD) to detect overloading problems, deviation from original plan, etc.

Upon detection of intolerable prediction/ planning errors, the real-time control algorithm tries to perform replanning based on the deviation (measured in real-time) from its original plan. This new plan aims to quickly compensate for this deviation, preferably without violating other requirements (such as electric vehicle (EV) charging deadlines).

The performance of the implemented real-time control algorithm is tested in simulation environment with data generated from a profile generator and the Triana network model of the test site of Lochem. Result shows that in most cases, the designed algorithm is capable of reducing the deviation from original planned power profile (sometimes at the cost of extending EV charging deadlines). In realistic cases (Case 3, Case 4 and Case 5), the real-time control algorithm tends to offer satisfactory improvement even compared with what is gained by the optimal solution (Table 4.6). It also helps to reduce overall power consumption upon detection of overloading problem.

Acknowledgments

Upon completion of my master thesis, firstly I want to thank my family for their unconditionally support throughout the last two years. Without their assistance I could not imagine finishing my master's study in a foreign land so far away from home.

During the last seven months at CAES and Locamation, I never cease to be amazed by the excellent academic achievements, professional work attitude and persistent pursuit of knowledge of all their members. Especially I am greatful to my supervisors, Prof.dr.ir. Gerard J.M. Smit, Gerwin Hoogsteen, Dr.ir. Omar Mansour, Dr.ir. Vincent Bakker and Prof.dr. Johann L. Hurink. Their guidance and generous help throughout the last seven months inspired me.

I also appreciate each course I took at University of Twente. The sparkling intelligence of all the teachers made all of them great enjoyment.

At last I wish to thank all my friends in the Netherlands. Wish all the people I know here can have the brightest future.

Contents

1 Introduction

1.1 Motivation

Energy has always been one of the driving forces promoting the advance of human society. In modern society, electricity plays an important role powering not only factories but also households. Traditionally, electricity has been produced centrally in large plants. It was transported through the high voltage (HV) grid down to the LV distribution network [9]. However, a trend in the coming decades shows that a larger part of electricity demand will be generated in distributed setting [2]. PV panels, wind turbines and micro-Combined Heat and Power (μCHP) emerge as distributed power generators, which are even possible to be installed in households. This new trend caters to the increasing demand of electric energy. But at the same time, it brings challenges. One of them is that since the old distribution network is not designed with DG like this in mind, the extra load may put the existing network in danger. Thus an improved version of the grid called *smart grid* is introduced . A definition of smart grid is given in [18]:

Definition 1.1 *A Smart Grid generates and distributes electricity more effectively, economically, securely, and sustainably. It integrates innovative tools and technologies, products and services, from generation, transmission and distribution all the way to customer devices and equipment using advanced sensing, communication, and control technologies. It enables a two-way exchange with customers, providing greater information and choice, power export capability, demand participation and enhanced energy efficiency.*

Smart grid technologies can be divided into three categories [2]: DG, Distributed Storage (DS) and DSM). Among them, research focusing on DSM technology has been conducted within CAES of the University of Twente for the last few years [2] [10] [14].

Definition 1.2 *DSM is the planning, implementation, and monitoring of utility activities designed to influence customer use of electricity in ways that will produce desired changes in the utility's load shape, i.e., changes in the time pattern and magnitude of a utility's load. [4]*

Triana is a three-step DSM control methodology developed within CAES for smart grids. With information from households and LV grids, it can be used to flatten the energy profile and improve power quality. In the first two control steps, it is able to utilize simulation or measurement data off-line to forecast and plan electricity production/ consumption on the device level. In the third step, a real-time control algorithm is applied to control each device at every time interval, ensuring the achievement of the plan. Triana simulator is an application developed within the CAES to realize this three-step control methodology. At the moment it supports simulation data and measurement data input.

However, sometimes when DSM methodology is applied, LV grid can still be subject to all kinds of problems. For example, DSM methodology aiming at reducing the overall energy bill may tend to converge controllable loads to low-price time. Whereas at the same time if uncontrollable loads behave unexpectedly from prediction, the overall power profile will deviate from its original plan. In an extreme case the LV grid might be overloaded. This overloading problem can lead to peak power demand and voltage drop, wearing the cables and bringing security problems.

Since prediction and planning errors can be caused by fortuitous behaviors of end users (like turning on many electric ovens within a short period of time or unexpected arrival time of EVs), it is often impossible to foresee it and prevent it from happening. However, it is possible to detect these errors and solve them using a real-time control algorithm. Therefore designing such a real-time control algorithm for Triana is made the main goal of this thesis.

To detect prediction and planning errors, power, voltage and current data of MV to LV transformers must be gathered in real time to be analyzed. At the moment, these data cannot be accessed and made use of in Triana yet. Thus making them accessible for Triana is targeted as another goal of this thesis.

To tackle this problem, this master thesis assignment is carried out within the CAES of the University of Twente in cooperation with Locamation. Locamation is a Dutch company which develops automation solutions for the smart grid. SASensor [1] [16], a substation automation platform within the MV and LV substations, is one of its products. It can continuously measure and monitor the electricity flow within the substation/transformers, making it a suitable choice for the source of RTD.

Now all measurements are available within ARTOS, a real-time operating system (OS) developed by Locamation. To allow customers of Locamation to utilize all those RTD, open platform V2 emerges. Avoiding running third party applications directly on ARTOS, it explores the possibilities to establish communications between applications running in parallel in different OSes. To be specific, by making use of KVM [6] [7] [11] [17], multiple guest OSes can be created and hosted by a Linux host. And one of the guest OSes is ARTOS, measuring and providing RTD available for third party applications running on host Linux or other guest OSes. Thus the remaining question is how to make RTD on ARTOS accessible for those applications, to be specific, for the Triana simulator in our case.

Under the premise of using KVM as the hypervisor, several inter-virtual machine (VM) data communication techniques can be exploited. Among them, a communication mechanism based on POSIX shared memory called Nahanni [12] was chosen by Locamation. It is free to be used commercially and it allows user-level data transmission between applications. Since Locamation is responsible to provide RTD from the ARTOS side for applications running in Linux to access, this thesis will mainly focus on Linux side running Triana simulator. For performance reasons, the Triana simulator is chosen to run on host Linux. Coupling ARTOS with the Triana simulator, the designed SASensor Open Platform V2 architecture is shown in Figure 1.1.

1.2 Task statement

To summarize, the main goal of this master assignment is to design a real-time control algorithm to determine and solve prediction and planning errors. It is the major objective of the designed system. Furthermore, an API must be added to the Triana simulator to access data residing in the shared memory as well. To be precise, upon detection of prediction and planning errors, the real-time control algorithm must react properly so that the following criteria are met.

- Upon detection, it must be ensured that overloading problem will be solved as soon as possible.
- The overall power profile remains as close to its original plan as possible.
- Charging deadlines of EVs set by end users are not violated.

Once necessary choice must be made among conflicting criteria based upon their priorities.

To prove the validity of the designed API and real-time control algorithm, the original plan is to apply it into the field test. For this purpose, the Lochem project is chosen.

Lochem is a Dutch village where its citizens form an union to generate renewable energy, accelerating the transition to renewable sources. As a result, PV panels are placed over the roofs of 50 houses and on the top of 30 municipal buildings [9]. IN4Energy [23] program is set up there where among others University of Twente and Locamation are involved. Lochem (IN4Energy project) is part of a national field test to test

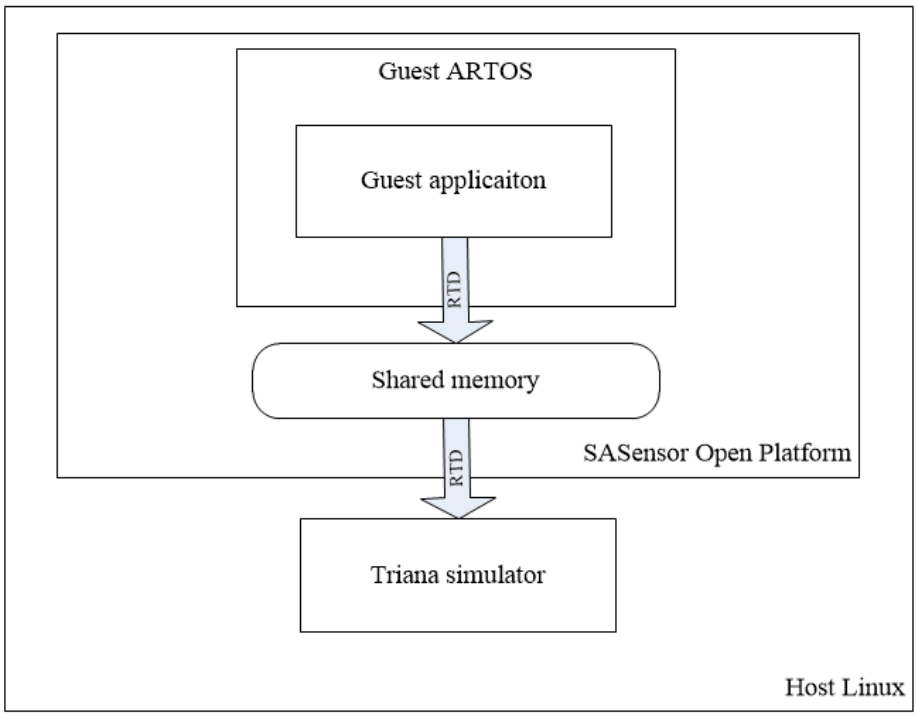

Figure 1.1: Designed SASensor Open Platform V2 architecture

the impact of high penetration of PV panels and EVs within LV grid. Furthermore, as discussed in [8], in two stress tests that have already been conducted there, large amount of EVs charing at the same time overloaded the LV grid, showing great impact on power quality. Because of those reasons, Lochem is considered a suitable test site for the real-time control algorithm.

The site of the first stress test in Lochem is shown in Figure 1.2. All three transformers and four feeders are metered and the power consumption of 83 households is known, among which 33 provide the Root Mean Square (RMS) voltage magnitude measurements.

However, due to time constraints the designed API and real-time control algorithm can not be tested in the field test. But since the Triana model of Lochem project is available, it is still possible to test if this additional functionality works as expected in simulation environment.

In the field test, the RTD are measured by sensors and sent to ARTOS. Hence in simulation a reliable substitute source of RTD must be found so that the control loop can be closed. The load-flow simulator discussed in section 2.3.2 provides a work around this issue as can be seen in Figure 1.3.

In load-flow calculation, households and fast charging poles in Figure 1.2 are modeled as nodes of the distribution network. In addition, resistance data of cables can be obtained from the power distributor. Suppose the power consumption of all nodes and the voltage levels of the secondary side of the MV/LV transformer in all three phases are known, it is possible to use load-flow calculation to determine the voltage levels at all nodes and current running through feeders. Those voltage and current information can form a steady state of the network.

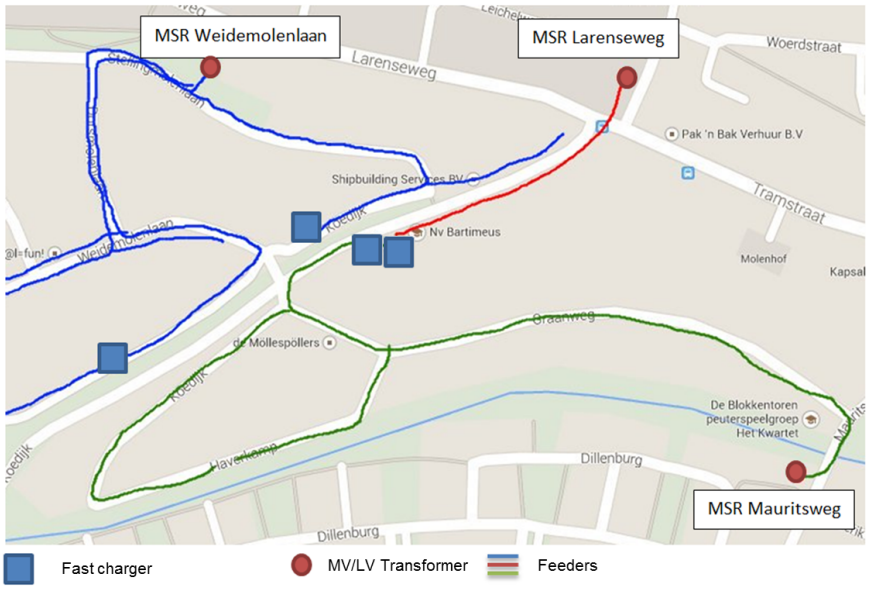

Figure 1.2: Test site in Lochem

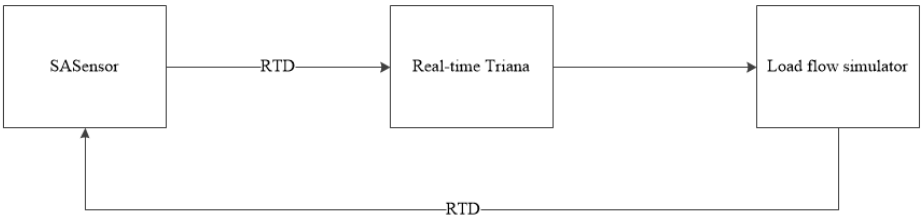

Figure 1.3: The control loop to test API and real-time control algorithm in simulation

However, to use the Triana model of Lochem project and the load-flow simulation as substitutions, their correctness must be ensured. In [8], they are verified using the information of the stress test conducted in Lochem. With the power profile of one of the feeders, the Triana network model and the location and measurement data of all fast charging poles, under several assumptions, the load-flow simulation shows the similar trend and voltage drop as measured in one of the meters, as shown in Figure 1.4 [8].

Some system errors might be included in the algorithm to perform load-flow calculations. Furthermore, the network model does not incorporate the effects of temperature, depth and surface type [8]. In addition, not all measurement data and location of households are available so assumptions and compromises must be made. However, from the observations of Figure 1.4, it is clear that despite all those errors, the Triana network model of Lochem project and the load-flow simulation can be trusted alternatives to test the designed API and real-time control algorithm in simulation.

1.3 Research questions

For the design of the real-time control algorithm, power, current and voltage are the main measurement data that can be analyzed and utilized in the Triana approach. Under the assumption that controllable

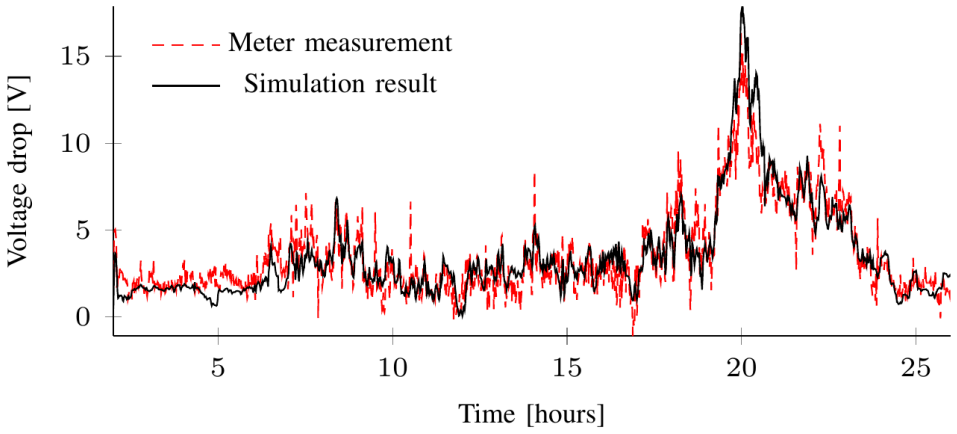

Figure 1.4: Voltage drop in the simulation and measured in one of the meters, taken from [8]

devices in real life can correctly execute their planned power profile, via controlling the functional states of those devices the real-time control algorithm can influence power more than other two factors. Thus it is reasonable to use it as the starting point while exploring the real-time control algorithm.

Let $P_{total}(t)$ be the overall power consumption at time t, at any state it can be divided into controllable part $P_{control}(t)$ (consumed by controllable devices), uncontrollable part $P_{no\_control}(t)$ (consumed by uncontrollable devices) and unknown part $P_{unknown}(t)$ (consumed by the cables, etc.),

$$P_{total}(t) = P_{no\_control}(t) + P_{control}(t) + P_{unknown}(t) \tag{1.1}$$

However since the duration of the simulation is set to one day which is way longer than that of the stress test, measurement data of the stress test can not be imported as the profile of the households (uncontrollable). Therefore there is no sufficient data to perform load-flow simulation. To work around this issue, in the simulation the profile generator developed by G.Hoogsteen (one of my supervisors) was in use. As a result, the load-flow simulator is excluded from the simulation and the control loop is simplified into what is shown in Figure 1.5.

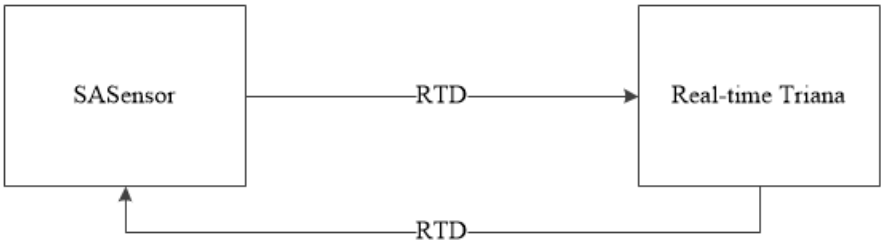

Figure 1.5: Simplified control loop to test API and real-time control algorithm in simulation

Since the profile generator can only generate $P_{no\_control}$, Equation 1.1 is simplified into Equation 1.2. However the omission of $P_{unknown}$ and the load-flow simulator brings only advantage, namely the

difference between $P_{unknown}$ in simulation and that in field test no longer exists.

$$P_{total}(t) = P_{no\_control}(t) + P_{control}(t) \tag{1.2}$$

Since in our use case, devices within households in Lochem cannot be steered (i.e. uncontrollable), EV fast charging pole(s) are the only subject(s) which can be controlled. Suppose there are N charging jobs in total, $P_{control}(t)$ is then equal to the total power consumption of all EV charging job(s) at all time.

$$P_{control}(t) = \sum_{i=1}^{N} P_{ev}(i)(t) \tag{1.3}$$

where $P_{ev}(i)(t)$ represents the charging power of the $i-th$ EV charging job at time t.

Let $P_{plan}(t)$ represent the overall power plan at time t. Based on equation 1.2 and 1.3, it can be concluded that when $P_{total}(t)$ deviates from $P_{plan}(t)$, the reason might be wrong prediction of uncontrollable power $P_{no\_control}(t)$ or unreachable plan of EV charging job(s) $P_{ev}(i)(t)$. At time t, this deviation $P_{dev}(t)$ can be represented by,

$$P_{dev}(t) = |P_{total}(t) - P_{plan}(t)| \tag{1.4}$$

When $P_{dev}(t)$ exceeds the preset margin, actions must be taken to correct $P_{plan}(t)$ in the future. Since the only control subjects are EV charging jobs, one or even all of them must be rescheduled so that the new overall plan $P_{plan'}(t)$ can approximate the original plan $P_{plan}(t)$ in the future.

In previous work, the modeling of LV network and household appliances, the prediction and planning of the Triana approach are well studied, implemented and tested. Therefore to reuse part of the original code base and algorithms of the Triana simulator can simplify the work of this thesis and improve its reliability. Some functionalities/ features might be reused are listed as follows.

- The planning algorithm of EV discussed in [5] and [24].
- The Triana network model of the test site of Lochem.

Because the planning algorithm and code of single EV is given, this thesis will focus on how to distribute $P_{dev}(t)$ to controllable loads so that $P_{plan'}(t)$ can remain as close to the original overall plan $P_{plan}(t)$ as possible.

Suppose a simple scenario with single household and three fast charging poles as shown in Figure 1.6. As in the Lochem case only the fast charging poles are controllable. When $P_{dev}(t)$ exceeds the preset margin, there are several possible reasons,

- $P_{no\_control}$ is lower than its predicted value $P_{no\_control\_pre}$. In this case, the charging power of available EV(s) can be increased such that the difference between $P_{plan}(t)$ and $P_{total}(t)$ is compensated.
- $P_{no\_control}$ is greater than its predicted value $P_{no\_control\_pre}$. Compared with the previous occasion, this one may cause the delay of EV charging jobs since in this occasion to stick to the original plan $P_{plan}(t)$, EV charging job(s) must decrease its charging power. It is also necessary to make a replan when $P_{no\_control}(t)$ drops back to or below its predicted value so that the charging deadlines set by end users can be met.

Besides prediction errors above, it is also possible to make planning errors for the EV(s). Although in the simulation it is assumed that the EV charging jobs can execute the power profile planned by device controllers without mistakes, unexpected late arrival time and early leave time still pose challenges to the real-time control.

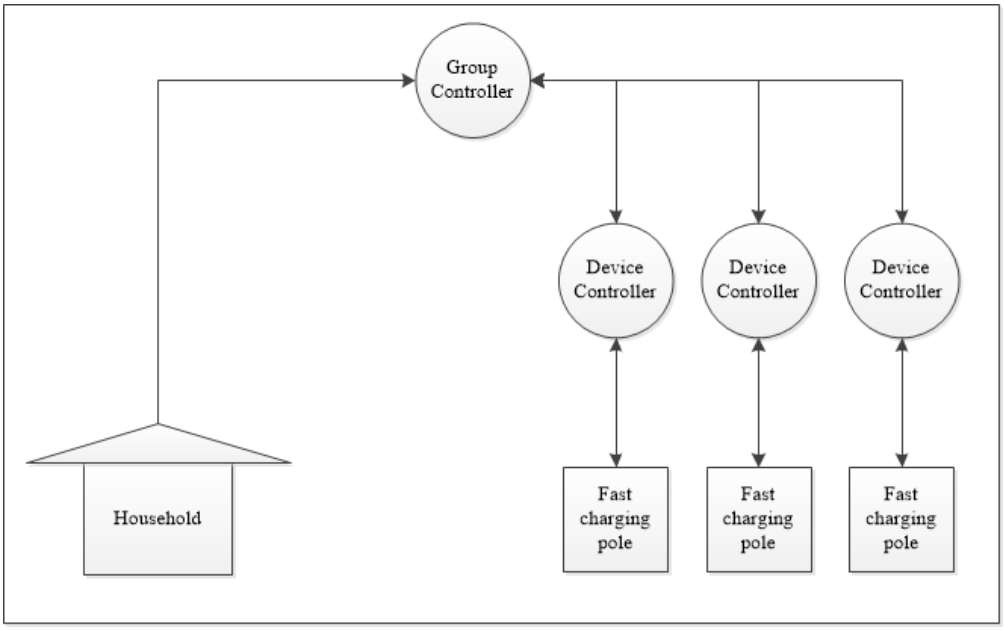

Figure 1.6: A simple example matching the scenario in Lochem

- Early leave time of EVs. In this case, $P_{total}(t) \leq P_{plan}(t)$. It is similar to the first scenario of prediction errors and the same strategy may also work in this case.
- Late arrival time of EVs. In this case, other available EV charging jobs can be replaned to make up for $P_{dev}(t)$ and when the new EV arrives, it can still meet its deadline utilizing the power "saved" by other early charging EVs.

Although there are more complicated scenarios, they can normally be solved by using the strategy of one of the cases mentioned above. For example, when overloading problem happens, the strategy used when $P_{no\_control}$ is greater than its predicted value $P_{no\_control\_pre}$ can be adopted. A more realistic case is when uncontrollable power peak shifts over time, which can be treated as the combination of the first two aforementioned scenarios.

In the aforementioned four simple scenarios, common challenges faced are which strategy to choose while performing the real-time control algorithm and how to make a replan for relevant EV charging jobs. Therefore my research question for the real-time control algorithm is,

- How to design the real-time control algorithm incorporated into the Triana simulator to realize the aforementioned objective of the system?
  - Which strategies to choose while performing the real-time control algorithm?
  - How to replan the controllable loads to keep $P_{plan'}(t)$ as close to $P_{plan}(t)$ as possible?

Also for the real-time control algorithm to access RTD, two major decisions must be made to design the RTD API. Each one of them forms a sub research question. Only by answering all of them can the best design of RTD API be found. For this part, my research question is:

- What is the best design for the API of the Triana simulator to access RTD?
  - Which measurement data from the ARTOS side has to be transmitted to run final simulation on the defined use case?

  - Based on Nahanni shared memory, what is the most efficient way to schedule shared memory access? (polling, interrupt, etc.)

1.4 Approach

The code base of the existing Triana simulator must be studied.

- To maximize reusability, information must be extracted from the model of EVs and the planning algorithm of Triana to determine which strategy is most suitable.
- It is necessary to figure out how to steer the selected device to make a proper replan for them.

In addition, a literature study will be conducted to answer the sub-questions listed above.

- Literature study can provide ample information about the data needed to detect prediction/ planning errors. This depends on the use case.
- To find the most efficient way to schedule memory access, polling, interrupt and other approaches must be compared as well. Pros and cons of each approach is available in literature.

1.5 Outline

What remains of this thesis is organized as follows. Chapter 2 gives a brief introduction on the background and related work of this thesis project. Then Chapter 3 answers the research questions by exploring the solutions of their sub-questions and discusses the implemented real-time control algorithm in detail. Followed are the simulation environment, results and analysis in Chapter 4. Finally a conclusion is drawn in Chapter 5.

2 Background and Related Work

Section 2.1 describes the source of RTD. How RTD is measured, put in ARTOS and then made available for the Triana simulator is included in this part. SASensor Open Platform V2 (section 2.1.1) enables communication between VMs via Nahanni shared memory (section 2.1.3) provided KVM (section 2.1.2) is utilized.

Followed is the description of Triana, both theoretical and practical. A brief introduction of how Triana approach is devised initially in theory is given at first. It is one of the theoretical background of the thesis. Then how the Triana approach is implemented as the newest simulator is described. It is essential to understand how it works to make modifications to it. A brief introduction of the concept of power peak shaving and how it is realized in the planning algorithm of Triana is described later. Flattening the power profile is one of the most important objectives of Triana.

Remaining topics are network modeling. Lochem project and its Triana network modeling are introduced. It is the use case of this thesis. At last, the additional module of the Triana approach, load-flow simulator is introduced briefly.

2.1 The source of measurement data

2.1.1 SASensor Open Platform

SASensor, as described before, is a substation automation system which is capable of performing various substation automation tasks. It can be further divided into SASensor High Voltage, SASensor High Medium Voltage (HMV) and SASensor Medium Voltage (MLV). For the remainder of this thesis, SASensor will refer to SASensor MLV platform installed within substation end.

A basic SASensor architecture can be composed of components listed below.

- Interface Modules, which can be subdivided into three types.
  - Current Interface Module (CIM), connected to current transformer for current measurements.
  - Voltage Interface Module (VIM), connected to voltage transformer to measure voltage.
  - Break Interface Module (BIM), which can sense the position of switch gears and actuate them when necessary. [1]
- Central Control Unit (CCU), which is capable of performing complex computations and controlling the behavior of BIMs.
- Versatile Communication Unit (VCU), connected to both the platform and remote control center, is responsible for remote operations and maintenances as well as substation information access.

Figure 2.1 shows all units mentioned above as well as their connections in a SASensor architecture example.

As depicted in Figure 2.1, all data measured by interface modules can be made available to CCU. CCU, running ARTOS, is able to resample, resynchronize and tag those asynchronous data with real time. All data is then stored as RTD variables in a special directory (/rtd) in ARTOS and handled by the dedicated RTD module. This allows fast data access by applications and supports all C data types except structures.

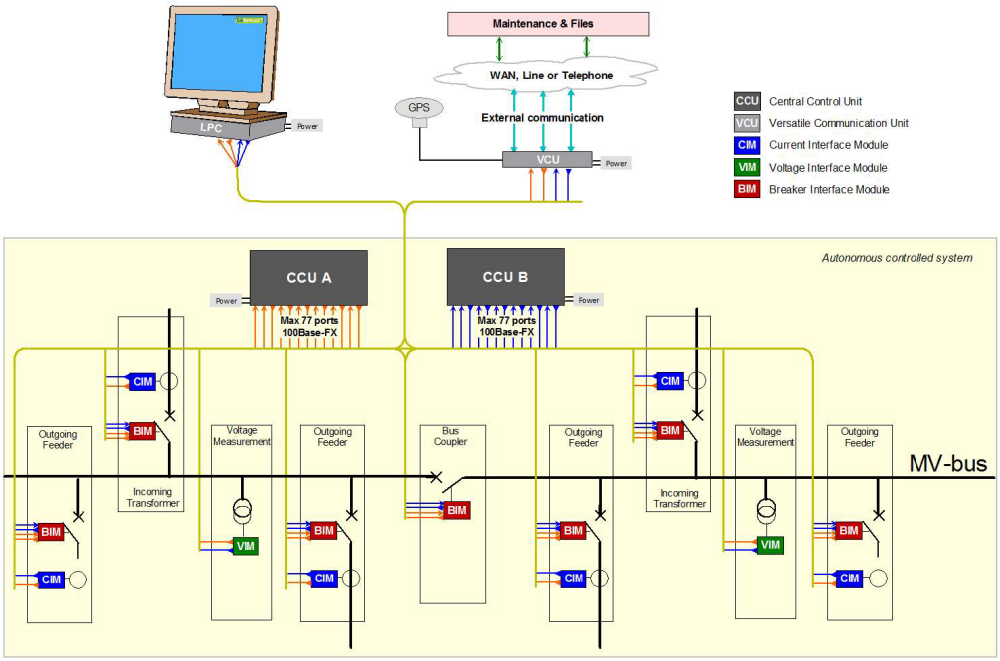

Figure 2.1: An example of SASensor architecture, taken from [16]

Currently the whole ARTOS is accessible via a web-based graphical user interface (GUI) which allows its user to navigate its file system hierarchically. However, efforts have been made to make SASensor an open platform for third party applications. The definition of Open Platform is given in [13],

Definition 2.1 *An Open platform is a SASensor automation system where clients (non-Locamation or 3rd party developers) can design and configure their substation and implement new ideas and functionalities in a non-proprietary manner.*

In SASensor Open Platform V1, several attempts are made to simplify the substation engineering process and reduce the dependency on the special expertise of Locamation's engineers. In [16], several ideas are initiated including developing new applications directly in C, translating Matlab/Scilab specifications to C, etc. However, all of them seem to bring platform-related restrictions and require certain knowledge of ARTOS. In [1], the notion of application store introduces a complex procedure of application development to protect (3rd party) Intellectual Property and ensure authority and role rules are enforced. But it still fails to solve problems aforementioned completely. However, getting rid of running third party applications on ARTOS, SASensor Open Platform V2 is a possible solution.

The concept of SASensor open platform V2 aims at providing better security, system reliability and richer set of tools for the 3rd party developers [13]. SASensor open platform V2 takes advantage of KVM by running ARTOS as one of its guest systems. Guest-guest and host-guest communications ensure the availability of RTD. Therefore, third party applications can run on its designed platform without compromise. In the following sections, the notion of KVM and how Nahanni shared memory allows inter-VM communication are introduced.

In the newest version of SASensor Open Platform, to eliminate the limitations of having both client applications and ARTOS on one physical system, a RTD server based on TCP/IP is devised. This expands

the capability of the open platform. However, in the context of this thesis, utilizing KVM and Nahanni is chosen because it can already accomplish the task of data communication under one physical system.

2.1.2 KVM

Virtualization is not a new concept in computer science. It allows multiple OSes to run concurrently as VMs on a single hardware platform. A Virtual Machine Monitor (VMM) or hypervisor, is a piece of software taking charge of resource virtualization of underlying hardware and concurrent execution of VMs [3]. An OS, on the other hand, is a piece of software brought together to perform common functions of controlling and allocating resources [19]. Noticing the functional similarities between VMM and OSes (scheduling, resource management, etc.), KVM is implemented as a kernel module which can be loaded to extend standard Linux, turning it into a VMM.

Besides the Linux kernel, KVM also relies on hardware support provided by hardware vendors (Intel and AMD) [6] [11]. Only by leveraging all those hardware support, can KVM exist alongside normal Linux kernel without interfering its code base. These supports also help simplify KVM into a small code base.

With all those supports, KVM can perform its duties properly. Compared with normal OSes, guest OSes in KVM differ in their ways of scheduling CPU power, managing memory and carrying out I/O operations.

- Under KVM, each virtual machine corresponds to a device node (such as /dev/kvm). These virtual machines, instead of scheduled by its own virtual CPU, are treated as normal processes and scheduled by host Linux scheduler.
- Host Linux kernel also helps to allocate memory to each VM. Virtual memory technology provides virtually contiguous blocks of memory to VMs although physically it is fragmented [7]. This kind of virtual-physical translations are recorded in the page table. In KVM, the introduction of VMs brings the need of an additional page table, which encodes the mapping of guest virtual memory to host physical memory. It is called shadow page table and encapsulated in kernel module (kvm.ko), providing vital support for memory management.
- As for the virtualization of I/O operations, KVM makes use of a modified QEMU, which is capable of emulating a whole personal computer (PC) platform including graphics, network cards and disk drivers. Corresponding to each running VM, a QEMU process is created. When I/O requests are made by guest OSes, they can be intercepted and redirected to be handled by QEMU process in user mode.

A KVM architecture example is shown in Figure 2.2. It can be seen that VMs are treated as user processes by the host. They normally run in guest mode unless sensitive instructions are trapped or I/O requests are made, where mode change to kernel mode or user mode must apply respectively. These mode changes may bring drawbacks to the performance of the designed system. As shown in the picture, I/O requests are handled by QEMU.

2.1.3 Nahanni

Since KVM project began later (2006) than other hypervisor projects (e.g. Xen), Inter-process Communication (IPC) research targeting at it is not as abundant as others. However, there are approaches explored to enable data transmission in this environment. Among them, Nahanni is chosen by Locamation for that it can be used freely for commercial purposes. Nahanni tries to provide a user-level shared memory interface for inter-VM communication on the premise of using KVM as hypervisor.

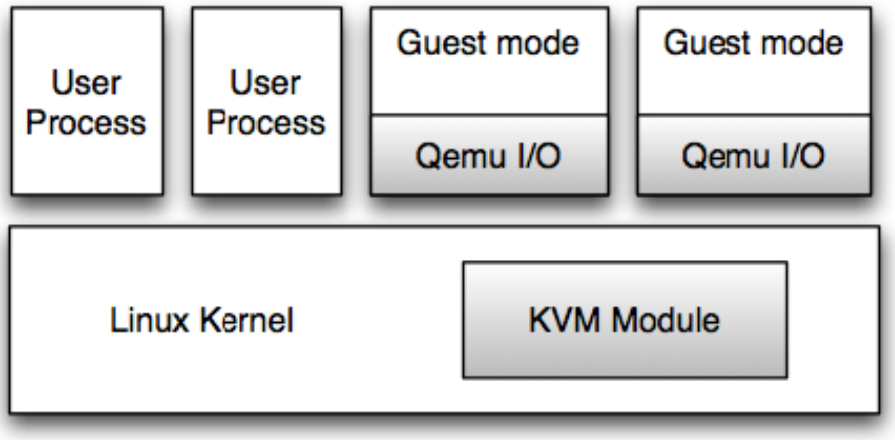

Figure 2.2: KVM architecture, taken from [17]

In [12], Nahanni is divided into three functional components, as listed below.

- POSIX shared memory. It already exists in Linux and no modifications to the Linux kernel are needed to make use of it. As discussed before, all I/O requests by guest OSes are handled by QEMU and each guest OS corresponds to its dedicated QEMU process. While Linux is able to create, modify and destroy shared memory region (shared-memory object) in an easy way, each shared memory region must be mapped into a QEMU process's user space in order to be accessed by its corresponding guest OS. Hence if a shared-memory object is mapped into the address space of multiple QEMU processes, their corresponding guest OSes are able to share data.
- Modifications to QEMU. The shared memory interface used in Nahanni is implemented as a standard device interface, providing support for unmodified OS to perform I/O operations on virtual devices the same way as it does on real hardware. To realize this goal, a new virtual Peripheral Component Interface (PCI) device called *ivshmem* is added to QEMU. This device, as other PCI devices, contains a configuration space, which is divided into parts to store information such as name, vendor ID and features. For ivshmem, three base address registers in this space are used to store important information for QEMU device driver to configure access for the device. In addition, to make allocated shared memory available for ivshmem, a method is added to update RAMblocks. RAMblocks is a structure used by QEMU to keep track of allocated memory. The last modification to QEMU is the command line support to add a ivshmem device while creating a new VM.
- ivshmem device driver. In our case, Locamation has provided a driver for ARTOS to utilize ivshmem device.

To broaden the application of Nahanni, its developers introduced a notification mechanism in addition to polling. Shared-Memory Server (SMS) is the host application to enable inter-VM interrupt mechanism. It is a centralized server running in another process on the host, managing all aspects of Nahanni [12].

2.1.4 Application

A typical way of applying SASensor Open Platform V2 is given in Figure 2.3. It is shown that voltage and current measurements can be gathered from the substation by Interface Modules and sent to ARTOS in the open platform. Those measurements are then processed, filtered and sent to third party application (e.g. the adaptive system model controller) running in real-time Linux. In this example the controller can apply those RTD into the real-time control of the EV charging poles.

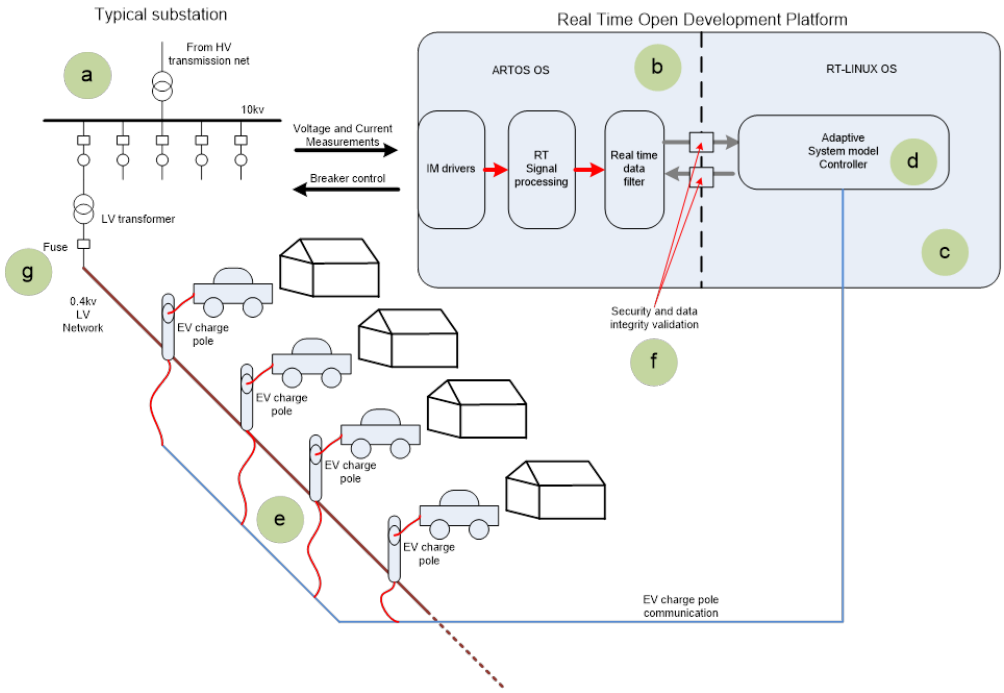

Figure 2.3: A typical application of SASensor Open Platform V2 (source: Locamation B.V.)

2.2 Triana

2.2.1 Triana approach in theory

Triana, as described before, can provide DSM solution for smart grid. In the concept of [14], each household/ building with its own smart/ non-smart electric devices can be modeled as a leaf node. These nodes, as other nodes, are equipped with controllers. Communication links are available between each node and its parent/ child node(s).

Triana consists of three control steps:

- In the first step, lowest (household) level prediction of energy production and consumption is made for the coming day and sent to the global controller. For each device, the predicted profile is based on various aspects like its historic usage and external factors such as weather.
- In the second step, the result of first step is regarded as input. Working toward a specified objective (e.g. peak shaving), the global controller of the root node determines steering signals for nodes one-level lower. Likewise, each of these nodes, with its own controller, uses steering signals received from its parent to determine steering signals for its child node(s). Finally, leaf nodes with their own building controllers can adjust their power profile according to the steering signals they receive. The adjusted profile can be sent upwards again. In an iterative process, steering signals are updated according to the adjusted profile, working towards the ultimate objective.
- In the third step, after a plan is made for individual leaf node in the second step, this plan (steering signal for each building) is treated as input. A real-time control algorithm is made use of in this step to execute the plan in a device level. To be more specific, for each device, constraints are introduced to define the valid states it can be. These constraints can be technical (e.g. related to the modeling of the device) or non-technical (e.g. catering to the demand of its user). Under those constraints, the

criterion to select the "best" state for each device is a cost function which expresses the desirability of all of its possible states, covering preferences of residents, wearing of the device, the global objective (steering signal for the building), etc. The weighted sum of all cost functions in one building is called optimization function, representing the desirability of a set of states chosen for all devices in the building. Therefore, to minimize the optimization function, the process of exploring the most desired state is simplified into an Integer Linear Problem (ILP) to minimize the cost function of each device. This ILP optimization problem can be extended with Model Predictive Control (MPC), as elaborately discussed in [14].

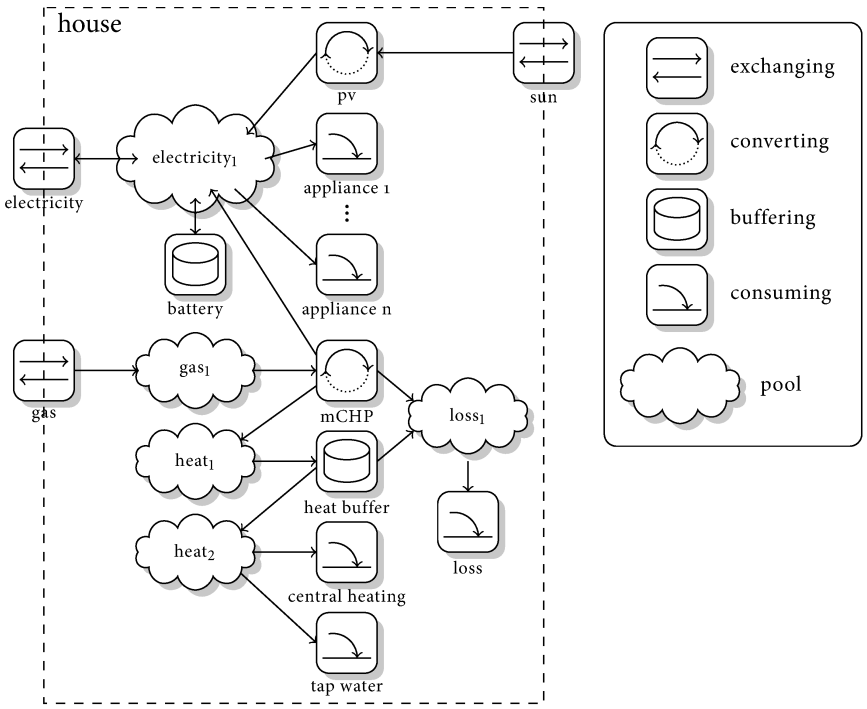

Figure 2.4: Basic house modeling in Triana [14]

To simulate the effect of this three-step control methodology, an application (Triana simulator) running on Linux is developed in CAES. Triana simulator accepts smart grid models written in Python scripts. The model, representing (part of) a smart grid, can contain households along with its devices, actual wire linking those buildings, control wires as well as energy streams. In our case, a fully established model from Lochem project is available. In [14], four different types of devices are created to model a household in Triana. They are buffering devices, consuming devices, exchanging devices and converting devices, as shown in Figure 2.4. Energy between devices is transfered by pools. Note that the actual symbolization of devices and pool may differ from that in the newest Triana simulator which is introduced later, however the basic modeling idea remains the same.

2.2.2 Work flow of existing Triana simulator

Before modifying the Triana simulator, its existing code base must be studied to maximize its reusability.

While developing the newest Triana simulator, compared with the aforementioned Triana three-step DSM methodology in theory, a few changes and compromises are made. They are listed as follows:

- In the Triana simulator, the first step, namely the local prediction is not performed. Generally speaking, smart devices capable of predicting are not widely installed in normal households throughout the Netherlands. (The same applies to the test field in Lochem as well.) Therefore the first step is omitted in the simulator. The Triana simulator takes one set of data directly as the input of the planning algorithm of the second step, the desired profile.

- In the Triana simulator, the second step can be executed much faster. This is because of the increase of computational power of PCs. Under this circumstance, the second step (i.e. planning) can also be executed within a short period of time (i.e. 15 minutes) like the third step (i.e. real-time control). Therefore a replanning is possible to be performed in the simulator every several time intervals (or even every time interval). This brings the possibility to merge the second and third steps together. The new planning algorithm is introduced in section 2.2.3.

- A new module called load-flow simulator is added into the Triana simulator. Its principle and function is discussed in section 2.3.2.

With these changes in mind, the work flow of the Triana simulator within one time interval can be divided into several control rounds, namely:

1. Group controller, all devices and load-flow simulator request control round(s) from the simulation controller. Group controller gets round #1; all basic devices (include EVs) get round #2; and load-flow controller gets the next round(s) to get ready and perform its duty.

2. In round #1, the group controller and device controllers work in an iterative way as described later to generate a profile for each controllable device. This control round does not necessarily present in every time interval though. How long a replan will take place depends on the planning horizon and replanning-related parameters of the group controller.

3. In round #2, devices carry out the plan.

4. In the next round, the load-flow controller calculates voltage, current information over the local grid to be used for further analysis.

5. In the last control round, a function of devices are called to store all kinds of information for display.

This work flow is also shown in Figure 2.5.

2.2.3 Power peak shaving

To understand the purposes and mechanisms of power peak shaving, what a power peak is must be introduced firstly. Power peak is a relative notion that needs a reference value [15]. This value often refers to a power limit that should not be exceeded. This limit could be based on constraints of the grid, preferences of end users and power distributors, etc.

Since power peak shaving can be performed at different levels (device level, household - and LV network -), its purposes normally vary. In household level, some power peak shaving mechanisms aim to react to the energy price schemes, avoiding being charged for peak usage. On the other hand, when applied in LV network, it can be used to control the overall power consumption within a predefined level for the safety of the grid. In the Netherlands, LV cables are buried underground and protected by a layer of paper. The fluctuations of current running through these cables can lead to certain thermal effects, some of which can cause permanent damages to this layer.

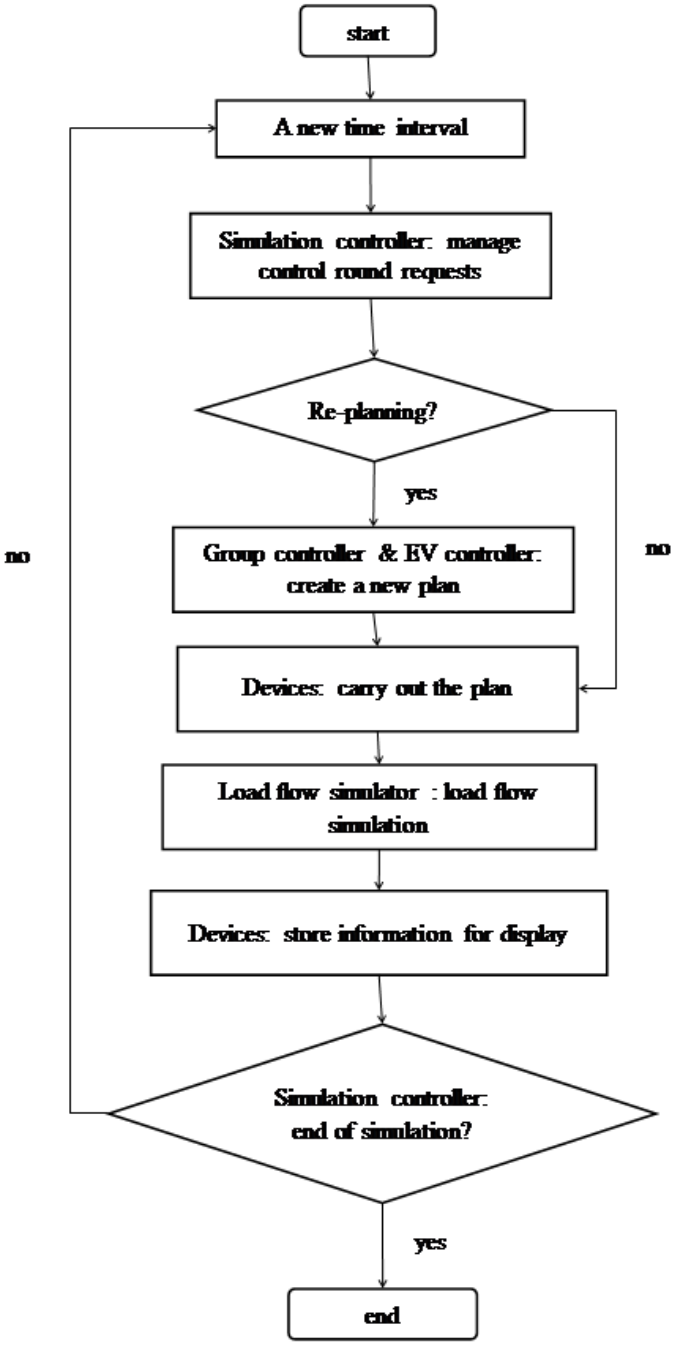

Figure 2.5: Work flow of the Triana simulator

Many mechanisms exist capable of "shaving" power peaks. In [15], a battery energy storage system is designed for power peak shaving applications. DSM methodologies such as Triana also support power peak shaving at different levels. In [21], for example, the controlling of a heat pump using Triana is studied. In Figure 2.6, the effect of power peak shaving can be seen in the electricity demand of the heat pump. At LV grid level, however, peak shaving in Triana is accomplished by approximating the preferred overall power

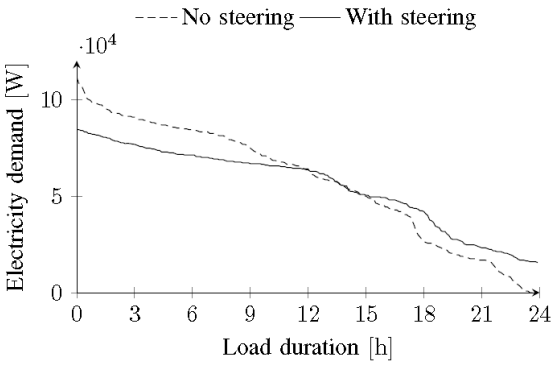

Figure 2.6: Power peak shaving performed in a heat pump case, taken from [21]

profile called desired profile. Often set to be flat, this desired profile is reached by minimizing the following formula,

$$Q = \sqrt{\sum_{t=0}^{End} (P_{plan}(t) - P_{des}(t))^2} \tag{2.1}$$

where $P_{des}(t)$ represents the desired global active power profile at time t.

This is realized in a hierarchical planning algorithm described in [5]. In this case, controllers, devices, pools and other elements are also modeled and organized hierarchically in the same way as described in section 2.2.1. The master controller is the root controller, which always has multiple child controllers. These child controllers can be either group controllers, which can have their own children or device controllers, which is the leave node.

Once the request of a new plan is given by the master controller, algirithm 2.1 is performed.

Algorithm 2.1 Hierarchical profile steering algorithm [5]

Master controller request each child $m \in \{1,\ 2,\ 3...,\ M\}$ to minimize its power profile $\|\overrightarrow{x}_m\|_2$
$\overrightarrow{x} \leftarrow \sum_{m=1}^{M} \overrightarrow{x}_m$ ▷ Total power profile
do
  $\overrightarrow{p} \leftarrow \{P_{des}(0),\ P_{des}(1),\ ...,\ P_{des}(End)\}$
  $\overrightarrow{d} \leftarrow \overrightarrow{x} - \overrightarrow{p}$ ▷ Difference vector
  for all $m \in \{1,\ 2,\ 3...,\ M\}$ **do**
    $\overrightarrow{p}_m \leftarrow \overrightarrow{x}_m - \overrightarrow{d}$
    For each child m find a new planning $\overrightarrow{\hat{x}}_m$ that minimizes $\left\|\overrightarrow{\hat{x}}_m - \overrightarrow{p}_m\right\|_2$
    $e_m \leftarrow \|\overrightarrow{x}_m - \overrightarrow{p}_m\|_2 - \left\|\overrightarrow{\hat{x}}_m - \overrightarrow{p}_m\right\|_2$ ▷ Relevant flexibility of m
  end for
  Find m with the highest contribution e_m
  $\overrightarrow{x} \leftarrow \overrightarrow{x} - \overrightarrow{x}_m + \overrightarrow{\hat{x}}_m$ ▷ Update total profile
  $\overrightarrow{x}_m \leftarrow \overrightarrow{\hat{x}}_m$ ▷ Update the profile of m
while $e_m < \epsilon$ ▷ Repeat as long as there is sufficient progress

Basically algorithm 2.1 tries to reduce the difference between P_{plan} and P_{des} in an iterative way. In each iteration, the master controller calculates the difference between current plan and desired profile at first.

Then it loops over all of its children in order to find a winner that can provide the most flexibility given its current plan. Once it is found, the winner will adopt its new plan. As a result P_{plan} is improved at the end of each iteration. This iterative process repeats until the convergence criteria are met. In this case, if the number of iterations is larger than permitted or the flexibility given by the winner in one iteration is lower than the preset value (ϵ), the iteration process terminates.

During the planning stage, an optimal global power profile is found which remains as close to P_{des} as possible according to the proximity factor Q in Equation 2.1. In this way, power peak shaving at LV network level is accomplished.

2.3 Network modeling

2.3.1 Lochem

As discussed in Chapter 1, to validate the designed architecture, it is applied into the simulation using Triana network model of Lochem. Previously two stress tests have been conducted there to study the effect of peak demand caused by high penetration of EVs and plug-in hybrid Electric Vehicles on the local LV grid. After each stress test a Triana network model corresponding to the test site settings was made and then validated with load-flow simulation [8]. Therefore in simulation the validity of Triana network model is guaranteed.

Figure 2.7 gives an overall view of the model. It can be seen that all households are connected to a master controller in the center by red wires, which represent control connections. Zooming in, it is clear that a cluster of red wires is actually the control connections between master controller and device controllers of individual household as shown in Figure 2.8. There is a LV network node in every household, represented by a lightening symbol. It has all kinds of voltage properties and is the gateway to local LV grid. Blue wires in the picture represent low voltage cables connecting households to local LV network. These blue wires as well as LV network nodes form a local LV grid, “powered” by a transformer which is also modeled as a LV network node. It is connected to a simulation controller symbolized with a chip in Figure 2.9. The simulation controller is used to simulate the network and manage all statistics.

In an ordinary household such as Figure 2.8, connected to LV network node is normally a three-phase exchanger (exchanging device), symbolized by two arrows. It can be applied to exchange energy between different parts or modules. Its another function is to exchange electric energy from energy stream simulations to load-flow calculations, suitable to model a smart meter for example. The yellow wires between LV network nodes and three-phase exchangers are energy streams, which can carry energy flowing from one device to another. In Figure 2.8, energy stream pool is connected to the exchanger. All domestic devices are interconnected through the pool. Furthermore, it can only transport energy with no loss. In other words, the amount of energy flowing in must be equal to the amount of energy flowing out at every time period. All kinds of devices (consuming-, converting- and buffering devices) can be connected to the pool. In the house model of Figure 2.8, a washing machine (modeled as a time shiftable device symbolized with a timer), an EV (modeled as a buffer time shiftable device symbolized with a battery) and an electric light (modeled as an uncontrollable device symbolized with a light bulb) are connected to the energy pool as an example. Washing machines, dryers and dish washers are normally modeled as devices which are capable of shifting consumption in time using static profiles. While EVs and heat pumps can be modeled as devices which can shift consumption in time using a buffer with state memory. Consumer electronics (e.g. lighting devices) are modeled as devices with static uncontrollable loads. Each device mentioned is connected to a Triana smart device controller based on its generic model. Each device controller is also connected to the central master controller to receive steering signals to perform its duty. Note the absence

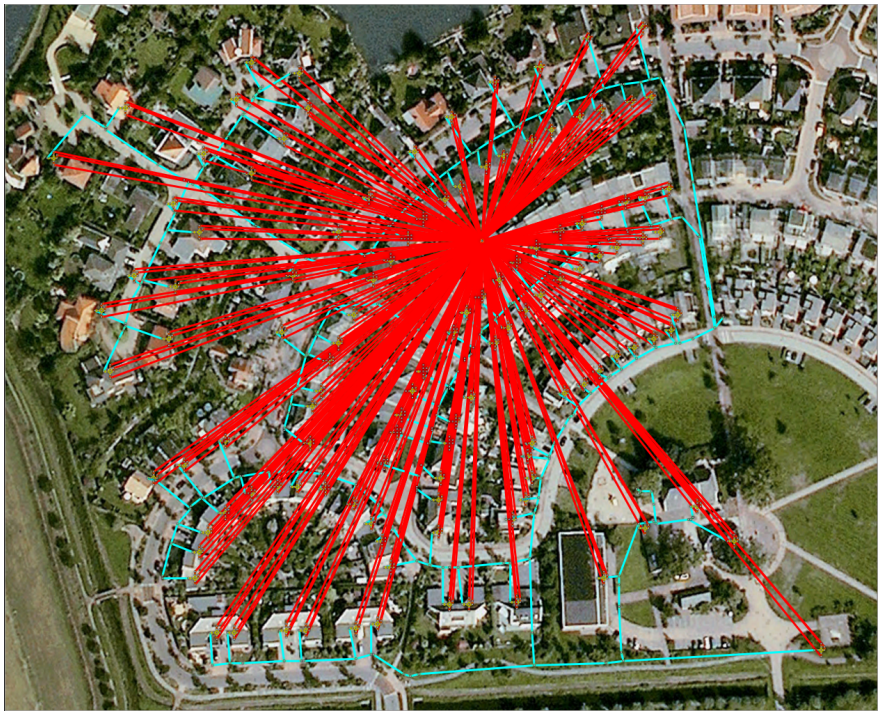

Figure 2.7: An overall view of the Triana model in Lochem project

Figure 2.8: Modeling of a single household in Lochem project

of buffering devices (e.g. heat buffer) and converting devices (e.g. μCHP) in this specific example shown in Figure 2.8.

Figure 2.9: Modeling of transformer and simulation controller in Lochem project

2.3.2 Load-flow simulation

The load-flow simulation module is added as a separate module in the Triana simulator. A definition of load-flow simulation is given in [9].

Definition 2.2 *Load-flow calculations on network models are used to obtain voltage levels, distribution losses and other network information for a certain scenario. These calculations are used in network design to validate that the required capacity will be realized.*

To use the results of load-flow calculations as the feedback of the Triana simulator, a separate module called load-flow simulator is implemented and added to the Triana simulator. The goal of the load-flow simulator is to determine all voltage levels, angles and power flows in the network. These information can form a steady state of the LV network containing the voltage level at LV nodes, the active and reactive power flowing in the network and the phase angle between voltage and current.

There exists multiple load-flow calculation algorithms. Newton-Raphson [20], Gauss-Seidel and Holomorphic Embedding Load-Flow method [22], to name a few. Among them, the Forward-Backward Sweep [26] is chosen and implemented [9]. It uses Kirchhoff's Current Law (KCL) and Kirchhoff's Voltage Law (KVL) and can be used for radial networks. In the Triana model, it works from the slack bus to the end nodes in the forward sweep, calculating the voltage at each node. The voltage levels are calculated based on the voltage drop along the branches. The voltage drop is caused by the impedance of the cables and the current running through them. Then in the backward sweep it works in the opposite direction to calculate the currents running in each cable. This process runs continuously until the convergence criterion is met. This is usually when the voltage levels calculated in two consecutive iterations are close enough.

Although the load-flow simulator can work alone, in the Triana simulator, it is coupled with the Triana approach. The interface between them is a function which sets the power consumption/ production levels of all nodes in the LV network. After all power levels are set the load-flow simulation then can be started. The results of load-flow calculation can in return be used as references when replaning.

3 Design Decisions and Real-time Control Algorithm

In the first part of this chapter, design decisions about the real-time control algorithm and the API of Triana to access RTD are made. Later based on those determined strategies, the designed algorithm is presented in section 3.2.

3.1 Design decisions

As listed in Chapter 1, it is essential to answer all sub-questions to find a best design for the real-time control algorithm and API. The Triana simulator has been proven to function as expected in many cases thus it is reasonable to use as much of its code base as possible.

As discussed in Chapter 1, three objectives of this real-time control algorithm are listed below.

- Upon detection, it must be ensured that overloading problem will be solved as soon as possible.
- The overall power profile remains as close to its original plan as possible.
- Charging deadlines of EVs set by end users are not violated.

In addition to these objectives, two other criteria are considered while exploring suitable real-time control algorithms.

- The modeling of the LV network and the planning step of Triana stay intact and act as the reference of the real-time control algorithm.
- If time allows, the algorithm should be extensible to be device agnostic.

With these in mind, when all sub-questions are answered, major design decisions are made at the same time.

3.1.1 Strategy explorations

In Triana, simulation time is divided into discrete time intervals, normally of length 1 *minute* or 15 *minutes*. Taking $15 - minute$ time base as an example, if an initial plan is made at the first time index (i.e. $interval[0]$) and the planning horizon of this plan is 96 indexes ahead ($96 * 15\ minutes = 1\ day$), for the following 95 indexes, Equation 3.1 can be checked to see if a replan should be performed.

Let α be the maximum degree of deviation allowed. When the following standard fails, the real-time control algorithm must react.

$$P_{dev}(t) \leq P_{plan}(t) * \alpha \tag{3.1}$$

Note that to avoid oscillation, Equation 3.1 can be extended to Equation 3.2 so that the real-time control algorithm is only triggered once P_{dev} is too large for M consecutive intervals, i.e.,

$$P_{dev}(t) > P_{plan}(t) * \alpha\ when\ T - M < t \leq T \tag{3.2}$$

where T is the current time interval during simulation.

However, Equation 3.2 can also be checked continuously (event-based control) instead of periodically (time-based -). Choices must be made between them.

Time-based control v.s. Event-based control
Although theoretically it is possible to design an event-based real-time controller instead of a time-based one, RTD streams are normally not continuous. In our case, RTD values are only available at certain time interval. Since events are based on those RTD, in this case they can only be triggered at a time base. Therefore their advantage of fast reaction no longer stands. Moreover, Triana itself is time-based as discussed earlier. In other words, to allow an event-based control algorithm, its original code base must be altered dramatically.

In addition, the most severe case that can happen and cause fatal damages to LV grid is overloading problem. However, The type of fuses used in the feeder of the test site is Jean Muller NH fuse-link. Its rated voltage is 400V AC. With maximum current of 160 A, its time-current characteristics are shown in Figure 3.1. When the feeder is overloaded it can stand current 25% more than its maximum current (200 A in this case) for hours. Therefore even in the most urgent situation it is not necessary to use a event-based

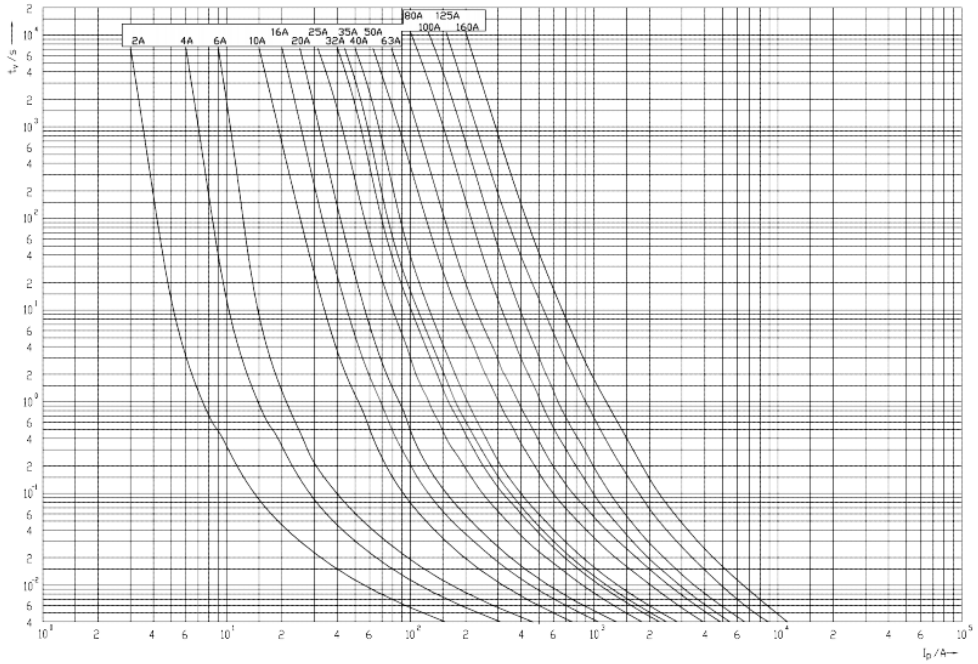

Figure 3.1: Time current characteristics of the fuses used in our use case (source: Jean Muller)

control algorithm.

Based on the reasons aforementioned, the control algorithm is set to be time-based. Since in simulation environment all RTD is available directly for Triana (Figure 1.5), every new control interval they are read and made use of instantly.

Desired profile v.s. Original plan
As described in Chapter 1 and at the beginning of this chapter, three conditions are made the target of the real-time control algorithm. Although solving overloading problem and no EV charging violation are posed by power distributor and end users respectively therefore rather non-negotiable, there are two major alternatives when it comes to which profile to approximate in the real-time control algorithm, namely the desired profile used initially in the planning step and the outcome of the planning algorithm.

Although theoretically it is possible to set a desired profile based on previous experience and history usage, in practice, normally in Triana it is good enough to use an empty profile as the desired profile of Algorithm 2.1. (In other words, $P_{des}(0) = P_{des}(1) = ...P_{des}(End) = 0$) If this case applies, the desired profile gives no further information at all.

Another reason is that Algorithm 2.1 can give an optimal solution minimizing Equation 2.1. Therefore approximating the original planning outcome is approximating the desired profile in some way. Not to mention that the desired profile is not necessarily accessible. For example, substituting P_{plan} with P_{des}, Equation 3.1 evolves into:

$$|P_{total}(t) - P_{des}(t)| \leq P_{des}(t) * \alpha \tag{3.3}$$

If the desired profile is too small at some point (e.g. the aforementioned empty profile), the real-time control algorithm will react much more often than necessary, even introducing oscillations.

Based on those reasons, it is chosen such that the actual overall profile should approximate the original plan (i.e. outcome of the planning step).

Replan v.s. Difference distribution
Before explaining these options in depth, four base scenarios mentioned in Chapter 1 must be analyzed at first. Suppose $T_{pre\_start}(i)$ and $T_{pre\_end}(i)$ represent the predicted start and end time of EV charging job i respectively and $T_{start}(i)$ and $T_{end}(i)$ symbolize the real start and end time of job i individually, each scenario can then be detected if its corresponding condition holds, namely,

1. $P_{dev}(t) = P_{plan}(t) - P_{control}(t) - P_{no\_control}(t)$, when the uncontrollable loads is lower than its prediction.

2. $P_{dev}(t) = P_{no\_control}(t) - (P_{plan}(t) - P_{control}(t))$, when the uncontrollable loads is greater than its prediction.

3. $T_{pre\_end}(i) > T_{end}(i)$, when EV i leaves earlier than expected.

4. $T_{pre\_start}(i) < T_{start}(i)$, when EV i arrives later than expected.

To compensate for the deviation in these cases, two types of strategies exist to be chosen from. The first one is replanning, meaning adjusting the original plan (choice of last subsection) according to the deviation and replanning with this adjusted profile as the desired profile. Another option would be to distribute whole or part of the deviation to one or even all of the available EV charging jobs (As described in Chapter 1, only EV charging pole(s) can be controlled in Lochem).

Although the second option seems to be able to compensate the deviation quickly, in the second case where the uncontrollable loads is greater than its prediction, this method will probably cause EV charging deadline violations since to make up for this deviation, EV charging power must be decreased. Therefore later to ensure each EV reaches its deadline, replanning is still necessary.

Additionally, as mentioned earlier in this chapter, even in the most urgent situation (overloading problem), the endurance of the grid allows the deviation to be compensated in a relatively long time. Therefore, replanning which can balance deviation from original plan against EV charging needs is considered a sufficient option here.

3.1.2 How to replan the controllable loads to keep $P_{plan'}(t)$ as close to $P_{plan}(t)$ as possible

When $P_{total}(t)$ satisfies Equation 3.2, there are four possible cases causing the deviation from original power profile. As discussed in Chapter 1, other scenarios can either be treated as the combination of those four scenarios or be solved by adopting the strategies of these cases. Hence in the remaining of this section those cases are discussed separately to find the most suitable way of replanning.

When uncontrollable load differs from prediction, it is necessary to adjust P_{plan} with P_{dev} to generate $P_{des'}$, the desired profile when replanning. The reason is that the deviation of uncontrollable load must be compensated in the new plan $P_{plan'}$; in other words the adjustment of $P_{des'}$ is the adjustment of the predicted uncontrollable profile ($P_{no\_control\_pre}$). For this reason, it can also be concluded when EV leaves earlier or arrives later than expected, P_{plan} can be used as $P_{des'}$ without adjustment since the uncontrollable load is the same as expected.

Moreover, when EV leaves earlier or arrives later than predicted, there is no need to check Equation 3.2 since via checking if $T_{pre\_end}(i) > T_{end}(i)$ or $T_{pre\_start}(i) < T_{start}(i)$, whether there will be an undesired deviation and it cause are already known.

Uncontrollable loads is lower than its prediction
In this case, the planning algorithm 2.1 can be reused. However, $P_{des'}$ must be adjusted according to P_{dev}. Suppose T is the time when Equation 3.2 stands, at this stage, since $P_{no\_control\_pre}(T)$, the predicted overall uncontrollable profile at time T is higher than $P_{no\_control}(T)$, from time T, the desired profile $P_{des'}$ can be,

$$P_{des'}(t_1) = P_{plan}(t_1) + P_{dev}(T) \tag{3.4}$$

where $t_1 \in \{T,\ T+1,\ T+2...,\ End\}$.

Equation 3.4 suggests the prediction that $P_{dev}(T)$ will last until the end of the simulation. However, this expectation generally does not stand. Thus parameter K is introduced which symbolizes the predicted length of drop of $P_{no\_control}$ from $P_{no\_control\_pre}$. Hence the extended $P_{des'}$ is,

$$P_{des'}(t_1) = \begin{cases} P_{plan}(t_1) + P_{dev}(T) & \text{if } t_1 < T + K \\ P_{plan}(t_1) & \text{if } t_1 \geq T + K \end{cases} \tag{3.5}$$

Equation 3.5 suggests the real-time control algorithm tries to compensate $P_{dev}(T)$ only for the following K time intervals. This strategy is illustrated in Figure 3.2 where $M = 4$ and $K = 5$.

In this example, from $interval[30]$ to $interval[33]$ the uncontrollable load is lower than expected for $M = 4$ intervals. Therefore at $interval[33]$, Equation 3.2 stands and replan is necessary. Since $K = 5$, from $interval[33]$ to $interval[37]$, P_{plan} is adjusted by $P_{dev}(33)$ to generate $P_{des'}$.

However normally the actual length of the drop of $P_{no\_control}$ is different from K. When K is greater than the actual length of the drop, no actions can be taken. While when K is too small, P_{dev} is only compensated until $interval[T + K - 1]$ (e.g. $interval[37]$ in Figure 3.2). For this reason, another replan with the same strategy at $interval[T]$ (e.g. $interval[33]$ in Figure 3.2) can be performed at $interval[T + K]$ (e.g. $interval[38]$ in Figure 3.2). This is illustrated in Figure 3.3.

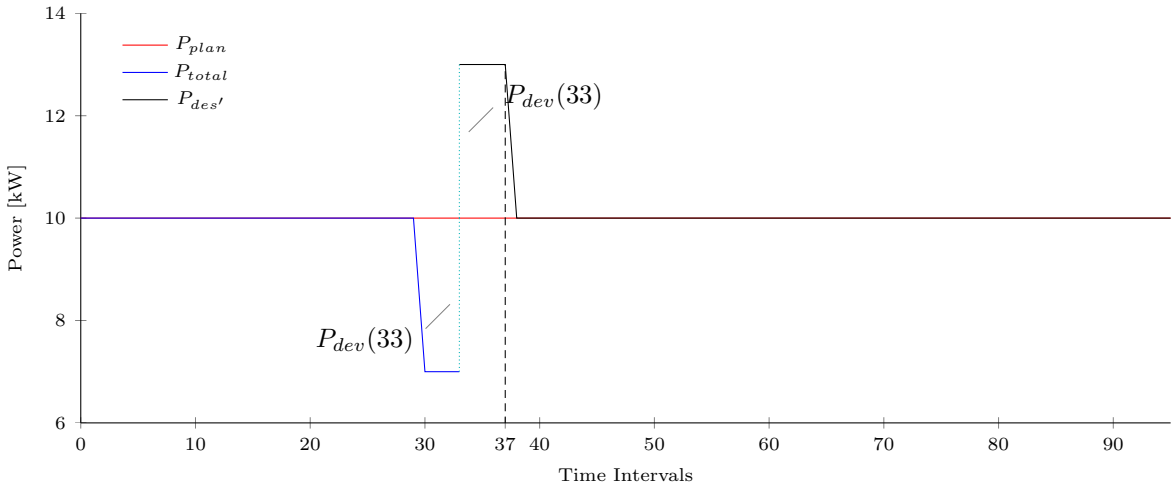

Figure 3.2: Illustration of the strategies adopted in Case 1 at *interval*[33]

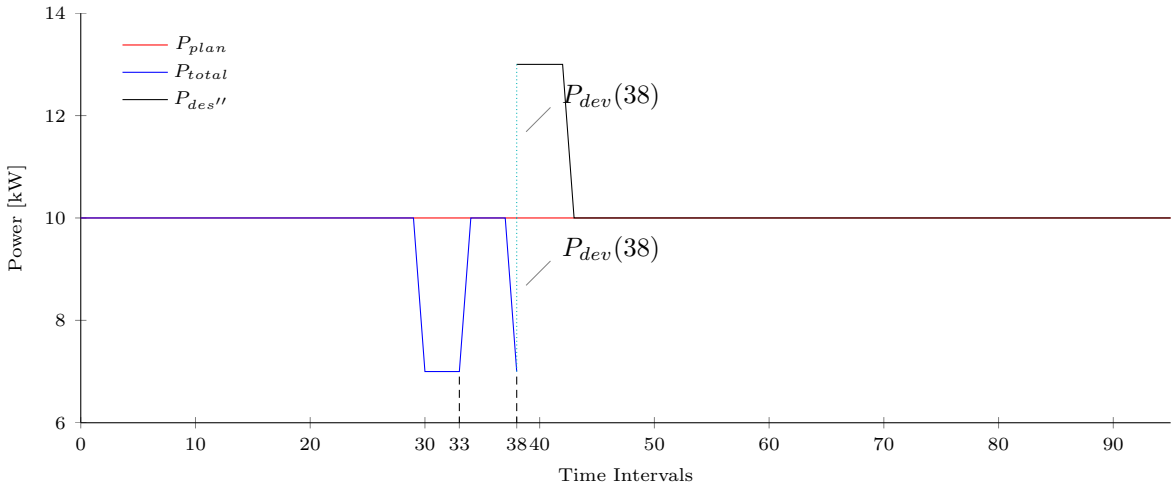

Figure 3.3: Illustration of the strategies adopted in Case 1 at *interval*[38]

To allow this fast reaction, F is introduced. It is a flag representing whether it is a lasting uncontrollable power drop/ peak. Once Equation 3.2 stands, F is set to $DROP$ (unpredicted uncontrollable power drop) or $PEAK$ (unpredicted uncontrollable power peak). Once $P_{no\_control\_pre} = P_{no\_control}$, F is set to its default value, $NONE$. The introduction of F means that even if K is much smaller than the actual length of the uncontrollable load deviation, it is still possible to compensate for it till its end via performing multiple replannings.

Uncontrollable loads is greater than its prediction

In this case, the strategy adopted is the same as that adopted in the last case. One exception is that Equation 3.5 should be replaced with Equation 3.6 because P_{dev} is the absolute difference between P_{total} and P_{plan}.

$$P_{des'}(t_1) = \begin{cases} P_{plan}(t_1) - P_{dev}(T) & \text{if } t_1 < T + K \\ P_{plan}(t_1) & \text{if } t_1 \geq T + K \end{cases} \tag{3.6}$$

As a result, the example in the last case evolves into what is shown in Figure 3.4 and Figure 3.5.

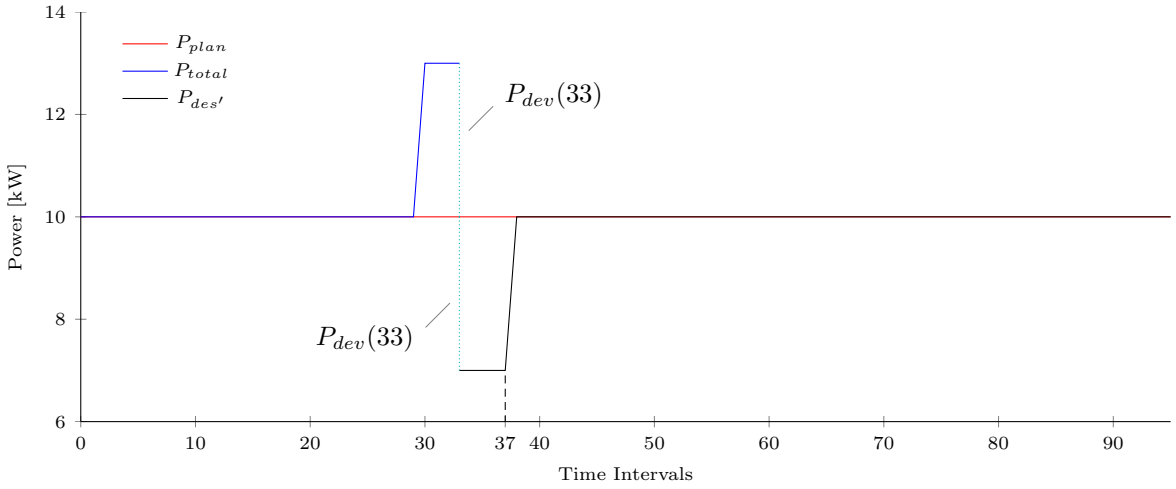

Figure 3.4: Illustration of the strategies adopted in Case 2 at *interval*[33]

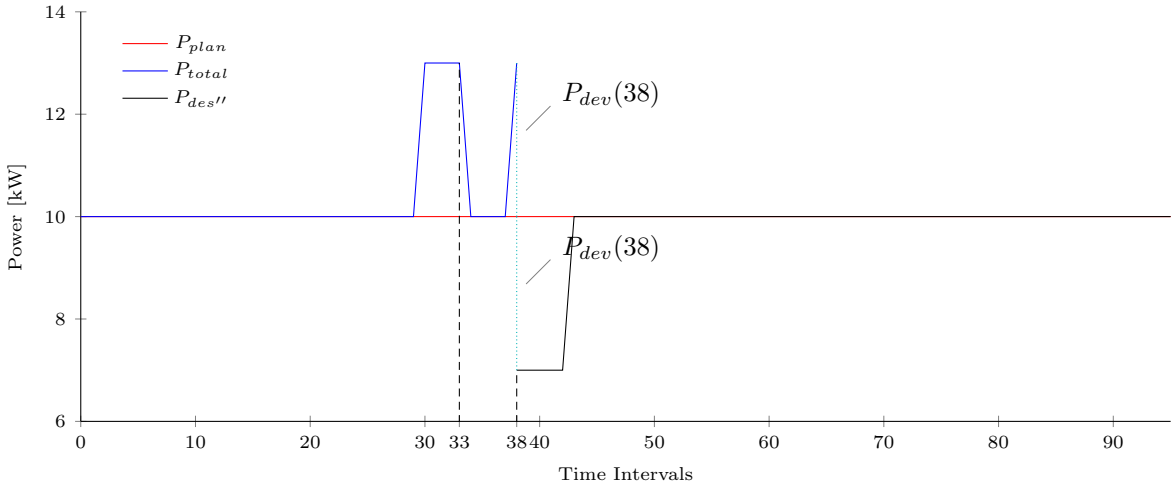

Figure 3.5: Illustration of the strategies adopted in Case 2 at *interval*[38]

However, different from the last case, when replanning is performed, the charging power of EVs is highly likely to decrease. Therefore, it might introduce charging deadline violations. To balance between obtaining a desired profile and charging EVs on time, Equation 2.1 in Algorithm 2.1 is extended to tolerate EV charging deadline violations,

$$\hat{Q} = \sqrt{\sum_{t=T}^{End}(P_{plan'}(t) - P_{des'}(t))^2} + \gamma * \sum_{i=1}^{N}(D_{ext}(i) - D(i)) \tag{3.7}$$

where there are N EVs in total, $D(i)$ and $D_{ext}(i)$ represent the charging deadline and the actual finish time of the $i-th$ EV respectively.

By extending Equation 2.1 with penalty for EV charging delay, during simulation it can be set whether it is preferred to suffer from EV charging deadline violations in exchange of a more flat overall profile. As a result, Algorithm 2.1 can not only be performed when every EV meets its deadline, but also be used when the delay $D \in \{1,\ 2,\ 3...,\ \infty\}$ is applied for each EV.

The strategies taken in this case can be concluded as Algorithm 3.1,

Algorithm 3.1 The strategies applied when uncontrollable load is greater than expected

1: calculate $P_{des'}$ according to Equation 3.6
2: Save all available EV charging jobs into a list L
3: Use Algorithm 2.1 with $P_{des'}$
4: Delay $D \leftarrow 0$
5: **while** $D \leq MAXIMUM\ DELAY\ ALLOWED$ **do**
6: $D \leftarrow D + 1$
7: **if** L is not empty **then**
8: Extend the charging deadline of each EV charging job in L by D
9: Use Algorithm 2.1 with $P_{des'}$, replacing step 9 of Algorithm 2.1 with Equation 3.8
10: Remove charging jobs making no contribution to Equation 2.1 from L
11: **else**
12: break
13: **end if**
14: **end while**

In Algorithm 3.1, the introduction of the maximum delay allowed for all EVs ($MAXIMUM\ DELAY\ ALLOWED$) and the list of all EV charging jobs (L) form two convergence criteria, namely,

- Delay D keeps increasing until $D \geq MAXIMUM\ DELAY\ ALLOWED$, Algorithm 3.1 ends.
- Every time D increases, the planning algorithm is performed again. After this planning process, if a EV charging job does not obtain a new plan (making no contribution to Equation 2.1), it is removed from L. When L is empty, Algorithm 3.1 stops.

In addition, for step 9 of Algorithm 3.1, when calculating the flexibility of each EV, penalty for deadline extension is included according to Equation 3.7. Therefore when using the planning algorithm, step 9 must be replaced with Equation 3.8.

$$e_m \leftarrow \|\overrightarrow{x}_m - \overrightarrow{p}_m\|_2 - \left\|\overrightarrow{\hat{x}}_m - \overrightarrow{p}_m\right\|_2 - \gamma * (D_m - \hat{D}_m) \tag{3.8}$$

where D_m is the delay of plan $\overrightarrow{x}_m$ and $\hat{D}_m$ is the delay of plan $\overrightarrow{\hat{x}}_m$.

Algorithm 3.1 can also be found as part of the real-time control algorithm in section 3.2.

EV(s) leave earlier than expected

In this case, the planning algorithm 2.1 can be reused for other EV charging jobs. Suppose EV j leaves earlier than expected, by design replan should only happen once when EV j actually leaves. In other words, at $T_{end}(j)$, a replan should be performed without EV j. Thus in the new plan, the charging power saved by EV j can be utilized by other EVs so that $P_{plan'}(t)$ can remain close to $P_{plan}(t)$. In this case, the original overall plan can be used as desired profile while replanning without being adjusted since the uncontrollable power profile is the same as expected.

EV(s) arrive later than expected
Suppose EV j arrives later than expected, when time reaches the predicted arrival time of EV j, the action performed when EV leaves earlier than expected can be reused. This means by design a replan without EV j should happen firstly at $T_{pre\_start}(j)$. Then at $T_{start}(j)$, EV j can then be made available and the second replan with all current EV charging jobs at the moment is necessary. Hence in theory, two replans are made in this case. The original overall plan can also be used as desired profile in both replannings without being adjusted.

3.1.3 Necessary measurement data

As described in Chapter 1, the major use case of this master thesis assignment is to detect and solve prediction/ planning errors. To inform the Triana simulator of this problem in time, RTD of transformers and feeders must be sent to it. This section will discuss which RTD is essential to fulfill this task. Since LV grid is the focus of the Triana simulator when monitoring prediction/ planning errors, only the outgoing field of transformers and the feeders connected to it will be metered in our case. The measurement points for one transformer are shown in Figure 3.6.

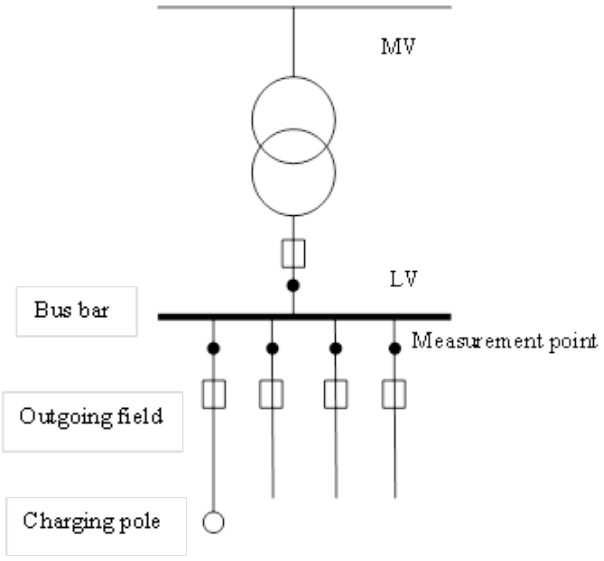

Figure 3.6: Measurement points for one transformer in this project

As shown in the picture, voltage of the secondary side of the transformer and current of all feeders connected to it can be measured. To be specific, the necessary RTD per phase for each measurement point is,

- power (active power)

In addition, line - neutral voltage of each phase must be provided as well if wanting to perform load-flow simulation.

For further analysis, it is also helpful to have the listed RTD per phase for each measurement point as well.

- power (reactive - and apparent power)
- phase angle
- current

In the Nahanni shared memory, each RTD is labeled with a unique "path" (e.g. /bay/01/current/L1). Based on those RTD, it is then possible to detect the problem in the LV grid as expected.

3.1.4 Scheduling memory access

Nahanni supports spinlocks which can monitor a value in shared memory [12]. Therefore, polling mechanism with primitives built upon spinlocks are clearly an option for synchronizing memory access between VMs. An alternative is to use interrupt to notify other VMs. As discussed in [12], the introduction of Shared-Memory Server makes it possible. Another inter-VM communication method often discussed is direct memory access (DMA). However, it is not possible to implement it in Nahanni given only a DMA engine. All possible options need to be compared to select the most suitable one.

Polling
Polling is the process where the computer or controlling device waits for an external device to check for its readiness or state, often with low-level hardware [25]. In our case, if an application tries to access a resource in the shared memory which is currently being used by another application in another VM, it will "spins" around the spinlock waiting until it is released so that the resource is free to access.

Interrupt
Interrupt, in [19], is defined as a hardware mechanism that enables a device to notify the CPU. In our case, this mechanism is useful when an application intends to notify other applications running in other guest OSes about the availability of resources in the shared memory. More details about the process of interrupt sending and receiving in Nahanni can be found in [12].

Comparison
As described in section 3.1.1, the transmission of RTD does not necessarily have to be frequent. Moreover, when to access the shared memory is known beforehand. Hence polling is considered a sufficient option in our case.

3.2 Real-time control algorithm

As discussed before in this chapter, there are four base cases acting as the target of the real-time control algorithm, namely,

1. Case 1, when the uncontrollable loads is lower than its prediction.
2. Case 2, when the uncontrollable loads is greater than its prediction.
3. Case 3, when EV(s) leave earlier than expected.
4. Case 4, when EV(s) arrive later than expected.

Based on the these cases and design decisions aforementioned, a real-time control algorithm is summarized. The only difference compared with what mentioned before is the introduction of β. It is the degree at which P_{dev} will be corrected. It is introduced to avoid oscillation.

Algorithm 3.2 A real-time control algorithm in Triana

1: At the start of a new time interval T(i.e. 15 $minute$)
2: Get RTD values via RTD module from the shared memory
3: **if** $P_{dev}(t) > P_{plan}(t) * \alpha$ when $T - M < t \leq T$ or $F == DROP$ && $P_{plan}(T) - P_{total}(T) > P_{plan}(T) * \alpha$ or $F == PEAK$ && $P_{total}(T) - P_{plan}(T) > P_{plan}(T) * \alpha$ **then**
4: **if** $P_{plan}(T) > P_{total}(T)$ **then**
5: $F \leftarrow DROP$
6: $P_{des'}(t) = P_{plan}(t) + P_{dev}(T) * \beta$ when $T + K > t \geq T$;$P_{des'}(t) = P_{plan}(t)$ when $t \geq T + K$
7: Perform the same planning algorithm with $P_{des'}$
8: **else**
9: $F \leftarrow PEAK$
10: $P_{des'}(t) = P_{plan}(t) - P_{dev}(T) * \beta$ when $T + K > t \geq T$;$P_{des'}(t) = P_{plan}(t)$ when $t \geq T + K$
11: Save all available EV charging jobs into a list L
12: Use Algorithm 2.1 with $P_{des'}$
13: Delay $D \leftarrow 0$
14: **while** $D \leq MAXIMUM\ DELAY\ ALLOWED$ **do**
15: $D \leftarrow D + 1$
16: **if** L is not empty **then**
17: Extend the charging deadline of each EV charging job in L by D
18: Use Algorithm 2.1 with $P_{des'}$, replacing step 9 of Algorithm 2.1 with Equation 3.8
19: Remove charging jobs making no contribution to Equation 2.1 from L
20: **else**
21: break
22: **end if**
23: **end while**
24: **end if**
25: **else**
26: $F \leftarrow NONE$
27: **end if**
28: **if** In Case 3, EV leaves or in Case 4 EV is expected to arrive **then**
29: Perform Algorithm 2.1 with all available EV(s) for the remaining planning horizon
30: **end if**
31: **if** In Case 4, EV actually arrives **then**
32: Perform Algorithm 2.1 with all available EV(s) for the remaining planning horizon
33: **end if**
34: Send new RTD values to ARTOS

4 Simulation Results and Analysis

In this chapter, the common simulation environment of all cases will be introduced at first. Followed are detailed settings of each case and the simulation results of each case. Based on the effect of the algorithm, analysis is made on each case separately. Finally, a general conclusion is drawn based on all matters above.

4.1 Use case

As discussed in Chapter 2, the Triana network model of Lochem is provided. As the stress test conducted on 2^{nd} of April, one of the four feeders in Lochem is tested. The transformer installed on Mauritsweg provides electricity for 71 households and three fast charging poles connected to this feeder. As mentioned in Chapter 1, the predicted uncontrollable profile is generated by the profile generator developed by G.Hoogsteen. In Figure 4.3 (red dotted line), it can be seen that the profile generated by it resembles what one normally expects from the total power consumption of dozens of households during weekend.

To read the power consumption of each household, an uncontrollable device controller/ reader is connected to each household, as shown in Figure 4.1. To plan and control EV charging jobs in each fast charging pole, a buffer time shiftable controller is connected to each of them. A master controller, the parent of all those controllers, is responsible for global planning and control. All control connections are shown in Figure 4.1.

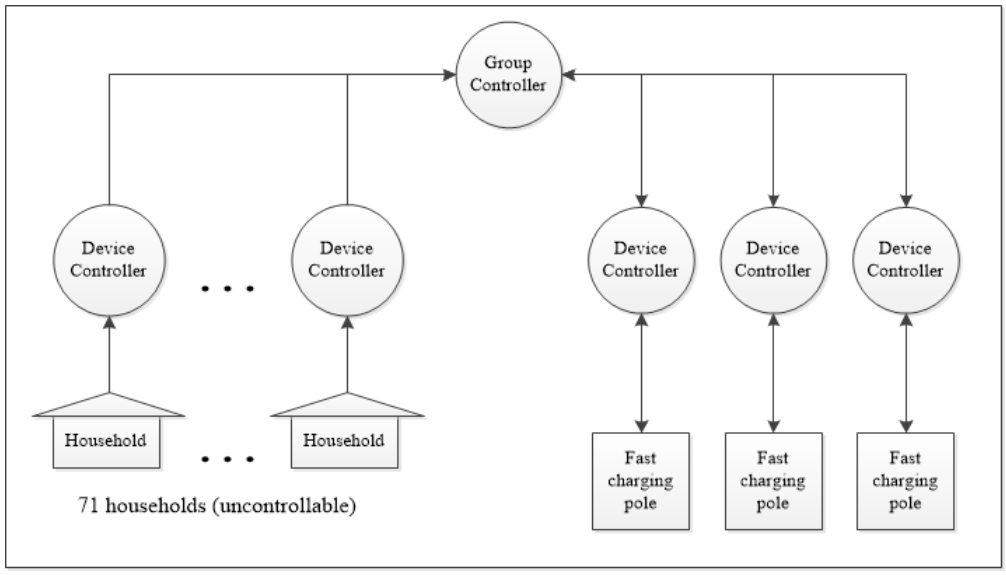

Figure 4.1: Illustration of Triana modeling and control connections of the use case

In the test environment, all EVs are set to be charged in three phases, meaning that when it is charged, it will consume the same amount of power in each phase. Therefore to avoid oscillations unrelated to the control algorithm, the master controller is set to be responsible for controlling all three phases and the uncontrollable power of all phases is set to be the same. This resembles a single-phase simulation. For the architecture shown in Figure 4.1, the algorithm can also function as expected if each phase is controlled by its own master controller and all EVs are single-phase charged despite the difference of uncontrollable profile in each phase.

The whole simulation process is divided into 96 consecutive time intervals. Since the duration of the simulation is one day, the time base of each interval can be calculated by,

$$Time\ base = 24\ hours/96 = 15\ minutes \tag{4.1}$$

In the simulation, at time interval 0, an initial plan is made for all devices. At every following interval, if Equation 3.2 stands, the real-time control algorithm discussed in section 3.2 will start to work. In the following section, the effect of this algorithm will be discussed in all four aforementioned cases as well as in a more realistic case where uncontrollable load shifts over time.

To allow quantitative measurement of the proximity of P_{total} with regards to the original plan, Equation 2.1 can be rewritten as,

$$Q' = \sqrt{\sum_{t=0}^{End}(P_{total}(t) - P_{plan}(t))^2} \tag{4.2}$$

where End represents the last time interval of simulation (e.g. 95 in the simulation).

In cases when EV charging deadline extension is involved, similar to Equation 3.7, Equation 4.2 can be extended into,

$$\hat{Q}' = \sqrt{\sum_{t=0}^{End}(P_{total}(t) - P_{plan}(t))^2} + \gamma * \sum_{i=1}^{N}(D_{ext}(i) - D(i)) \tag{4.3}$$

where there are N EV charging jobs in total. With Equation 4.2 and Equation 4.3, it is then possible to evaluate the performance of the real-time control algorithm by comparing $\hat{Q}'$ between with and without Algorithm 3.2 with Equation 4.4,

$$Improvement = (\hat{Q}'_{no\_rtc} - \hat{Q}'_{rtc})/\hat{Q}'_{no\_rtc} * 100\% \tag{4.4}$$

where $\hat{Q}'_{no\_rtc}$ represents $\hat{Q}'$ when Algorithm 3.2 is not introduced and $\hat{Q}'_{rtc}$ is $\hat{Q}'$ when the real-time control algorithm is performed. With Equation 4.4 it is possible to calculate the improvement of $\hat{Q}'$ brought by Algorithm 3.2.

Although theoretically it is possible to achieve 100% improvement of $\hat{Q}'$ in Equation 4.4, in cases where the energy consumed by P_{plan} differs from that consumed by P_{total}, the maximum improvement is lower. Suppose $P_{optimal}$ is the overall power profile that realizes maximal improvement, it can then be gained by evenly spreading $\sum_0^{End} P_{plan} - \sum_0^{End} P_{total}$ from the first interval when P_{total} differs from P_{plan} to the end of the simulation (e.g. $interval$[95] in our case), as illustrated in Figure 4.2.

In the example shown in Figure 4.2, from $interval$[30], P_{total} starts to deviate from P_{plan} till $interval$[50]. The optimal solution as described before would spread the total amount of deviation to the remaining simulation time (i.e. $interval$[30] to $interval$[95]), shown as $P_{optimal}$ in Figure 4.2. As demonstrated in Figure 4.2, the integral of P_{total} is the same as $P_{optimal}$, which proves that under the assumption that EV charging deadlines and the shape of uncontrollable loads are not taken into account, $P_{optimal}$ is a feasible solution.

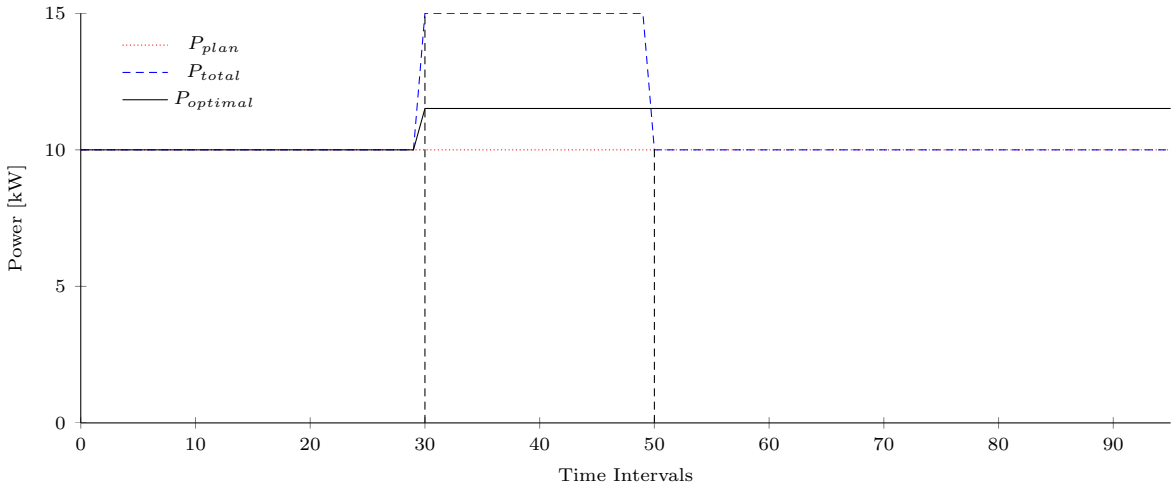

Figure 4.2: Illustration of the optimal overall power profile achieving maximal improvement

4.2 Results and analysis

4.2.1 Case 1: uncontrollable load is lower than predicted

	(Predicted) Start time	(Predicted) End time	Max charging power[kW]	Capacity and Setpoint[kWh]
EV 1	45	60	17.1	30
EV 2	50	80	14.1	66
EV 3	50	70	17.1	51

Table 4.1: Detailed information of all EV charging jobs

The detailed settings of EV charging jobs in this case can be found in Table 4.1. In order to create an unexpected uncontrollable power drop, from time *interval*[50] to *interval*[70], the real uncontrollable load is 0.65 times the predicted uncontrollable load in all three phases. This is shown in Figure 4.3.

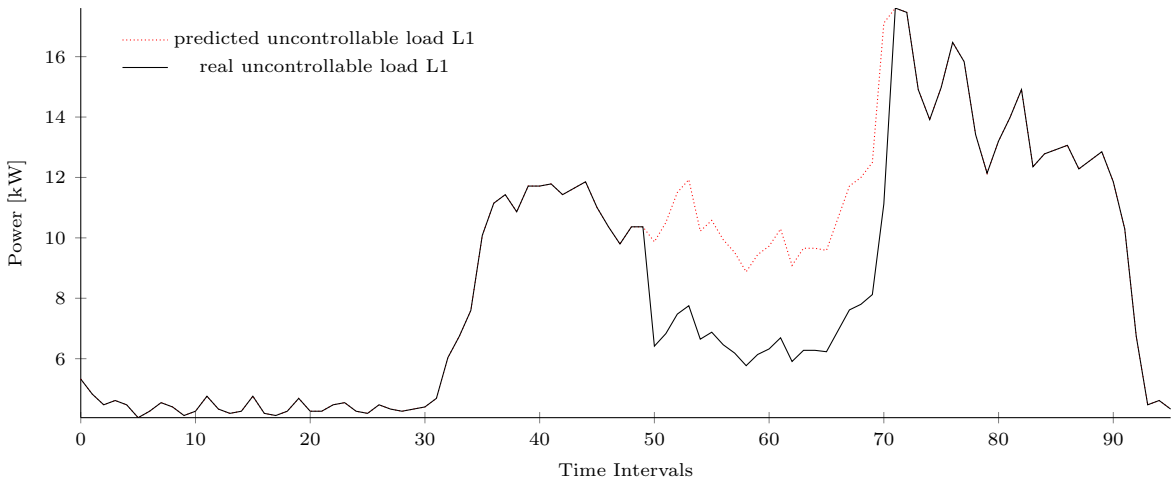

Figure 4.3: Predicted uncontrollable load and real uncontrollable load of phase 1

In the experiment, before *interval*[54], the real-time control algorithm never performs replanning. At *interval*[54], due to the changes in Figure 4.3, Equation 3.2 stands ($M = 4$). Therefore the 1^{st} replanning is performed. This new plan, according to Algorithm 3.2, tries to compensate for the next 5 intervals ($K = 5$) the deviation caused by wrong prediction of uncontrollable loads. As a result, after 5 intervals, Equation 3.1 is violated again at *interval*[59]. It is then decided that the length of the deviation of uncontrollable loads is longer than expected (K). Thus the 2^{nd} replanning with the same strategy as the 1^{st} one is performed. For the same reason, the 3^{rd} replanning is done at *interval*[64], i.e. 5 intervals after the last replan. These replan times can also be found in Figure 4.4.

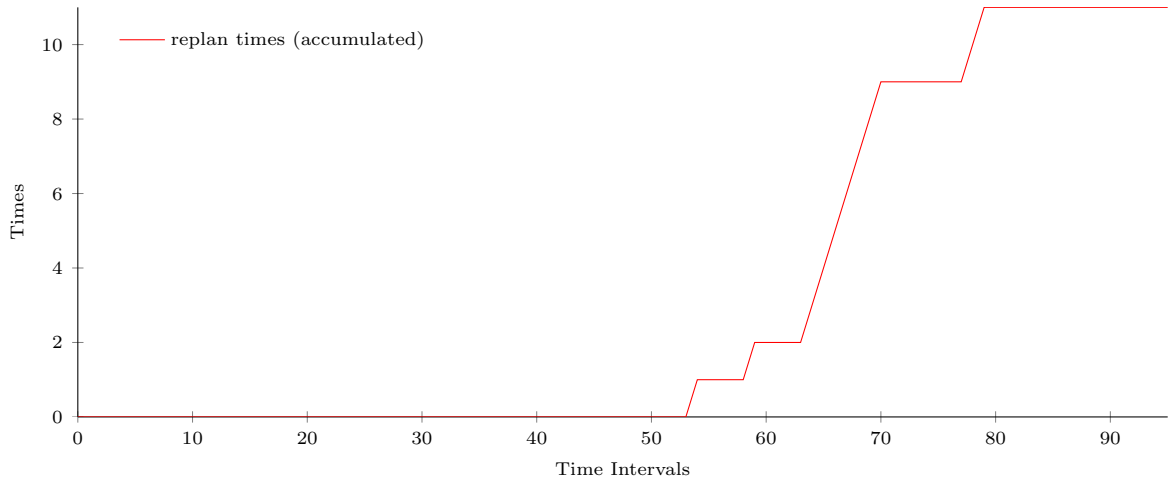

Figure 4.4: How many times replan has taken place (accumulated)

Then at *interval*[65], EV 3 is almost fully charged ($46121.3/51000 = 90.4\%$), as can be seen in Figure 4.5. Thus the potential of it to compensate the "missing" part of uncontrollable load is limited. This is when the first oscillation happens. From *interval*[65] to *interval*[70], Algorithm 3.2 fails to compensate this decrease of uncontrollable load at every time. Therefore replan times keep increasing as shown in Figure 4.4.

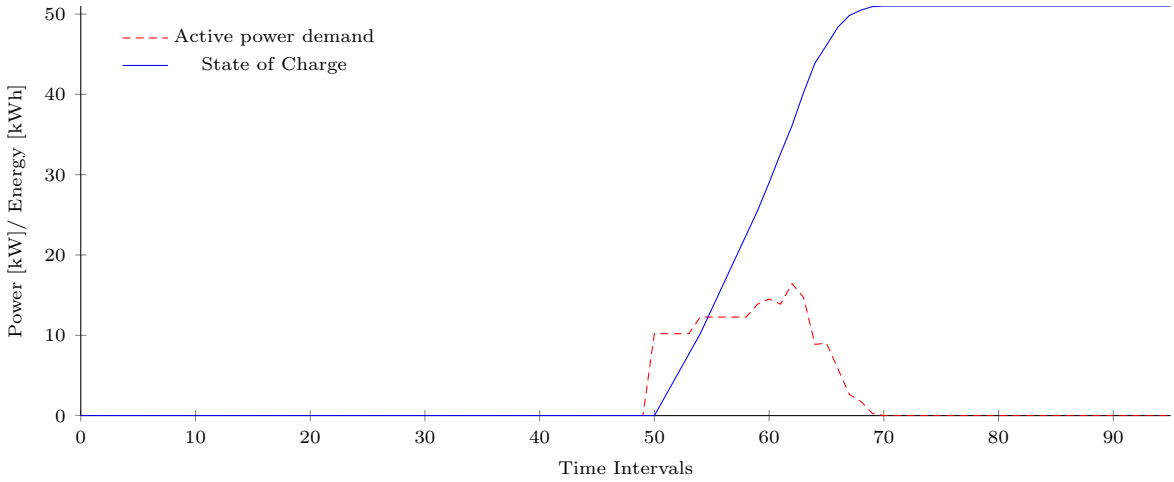

Figure 4.5: Active power demand and state of charge of EV 3

Then at *interval*[78] and *interval*[79], since EV 2 has also finished its charging (its deadline is 80 but it finishes earlier at *interval*[75] as shown in Figure 4.6), the new plan is much lower than the original plan (Figure 4.7). However, even though at both intervals Equation 3.1 is violated, nothing can be done because no controllable loads are at disposal.

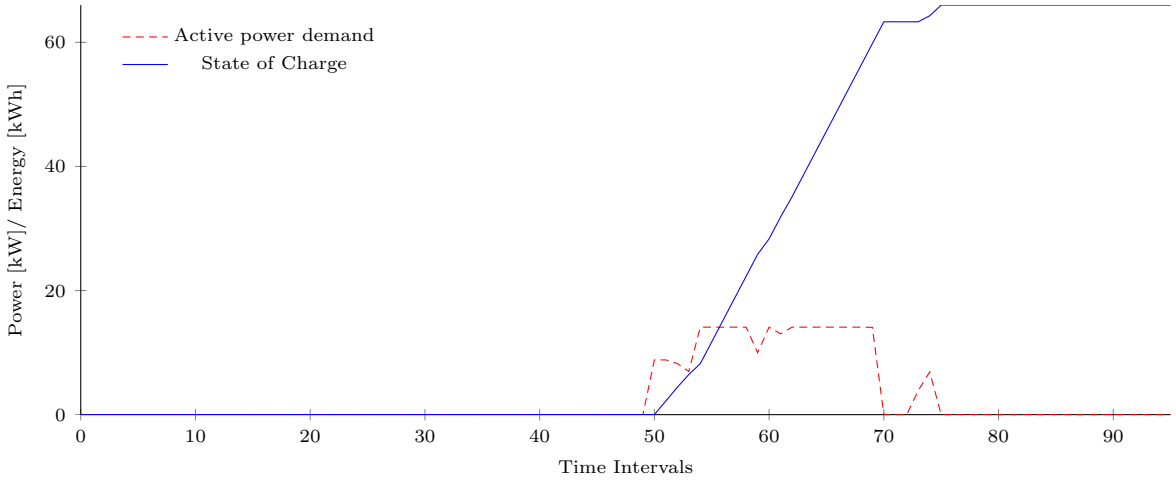

Figure 4.6: Active power demand and state of charge of EV 2

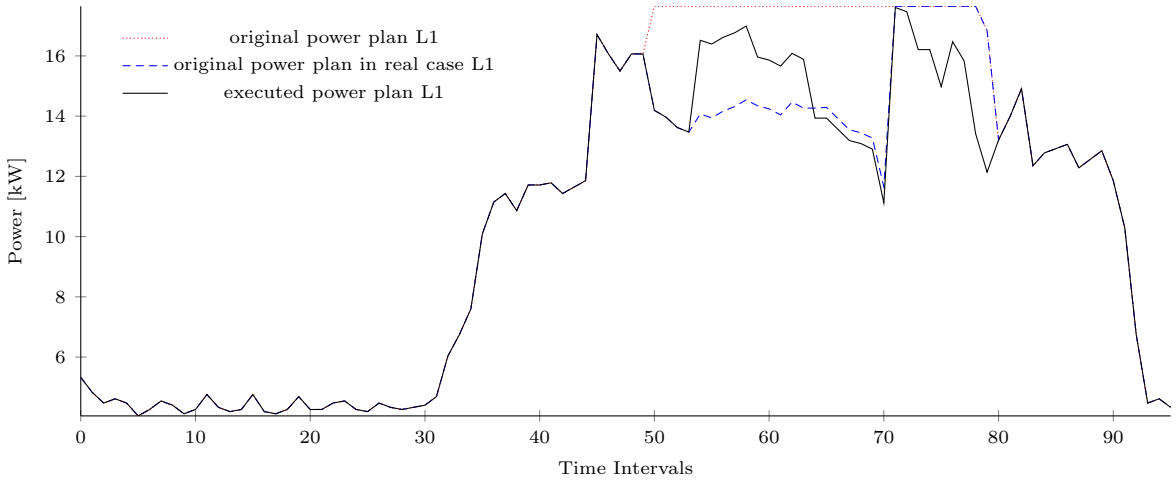

Figure 4.7: Original plan (predicted and real life) and final executed plan of phase 1

In Figure 4.7, it is shown that compared with taking no actions (blue dashed line), the new plan (black solid line) stays more close to the original plan (red dotted line). This can be proved using Equation 4.4. It can be calculated that the new planning can improve the proximity factor Q' by 2.7% $((17363.70 - 16898.71)/17363.70)$. The improvement is quite limited compared with the optimal plan $((17363.70 - 11578.32)/17363.70 = 33.3\%)$ obtained using the method depicted in Figure 4.2. This is mainly because from *interval*[54] to *interval*[64], the charging power of EV 3 is exhausted. When it is almost fully charged at *interval*[65], it can no longer help to improve the proximity factor Q', leading to oscillations until *interval*[70].

Furthermore, EV 2 is fully charged at *interval*[75], thus from *inteval*[75] to *interval*[79], Equation 3.1 is violated for 5 consecutive intervals (while $M = 4$). Since nothing can be done to compensate this deviation at *interval*[78] and *interval*[79], another oscillation happens. In this case, all EV charging jobs are finished on time though. Overloading test is not included in this case.

4.2.2 Case 2: uncontrollable load is higher than predicted

	(Predicted) Start time	(Predicted) End time	Max charging power[kW]	Capacity and Setpoint[kWh]
EV 1	45	60	17.1	30
EV 2	50	80	14.1	36
EV 3	50	70	17.1	30

Table 4.2: Detailed information of all EV charging jobs

The detailed settings of EV charging jobs in this case can be found in Table 4.2. In order to create an unexpected uncontrollable power peak, from time *interval*[50] to *interval*[70], the real uncontrollable load is 1.4 times the predicted uncontrollable load in all three phases. This is shown in Figure 4.8.

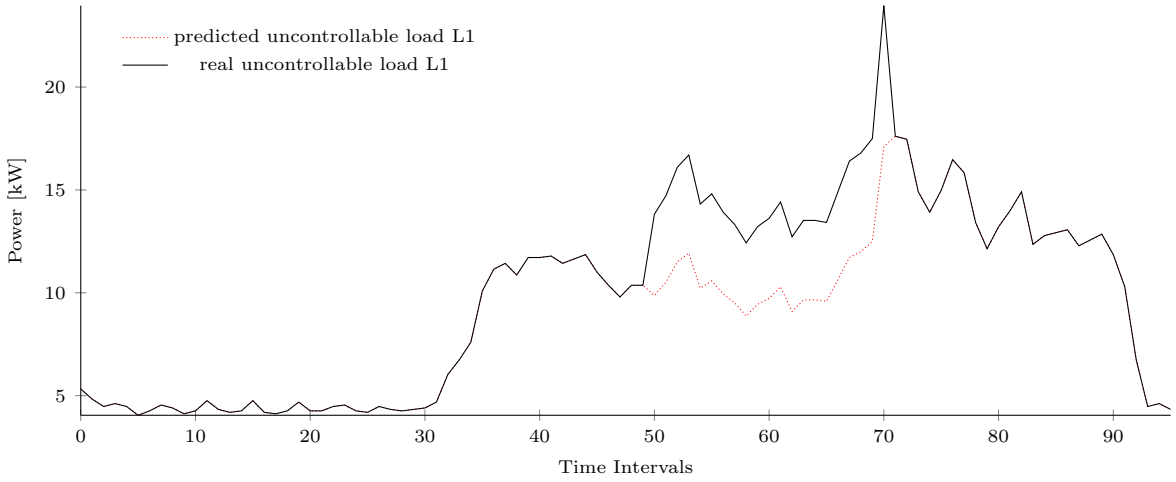

Figure 4.8: Predicted uncontrollable load and real uncontrollable load of phase 1

In order to test the effect of γ, the penalty coefficient in Equation 3.7, two experiments are conducted with the same settings above. Maximum deadline extension is set to 10 intervals in this case.

$\gamma = 50$
In the 1^{st} experiment, $\gamma = 50$, meaning that the penalty for EV charging delay is small (compared with $\gamma = 300$ in the 2^{nd} experiment). As a result, if enough improvements can be gained by approximating the original power profile, charging deadline(s) might be extended. This can be observed in Figure 4.9 and Figure 4.10. While EV 1 is charged on time, the deadlines of EV 2 and EV 3 are extended by 10 intervals to *interval*[90] and *interval*[80] respectively. Under this condition, the proximity of the new plan to the original plan can be improved.

In the experiment, before *interval*[53], the real-time control algorithm never performs replanning. At *interval*[53], due to the changes in Figure 4.8, Equation 3.2 stands ($M = 4$). Therefore the 1^{st} replanning

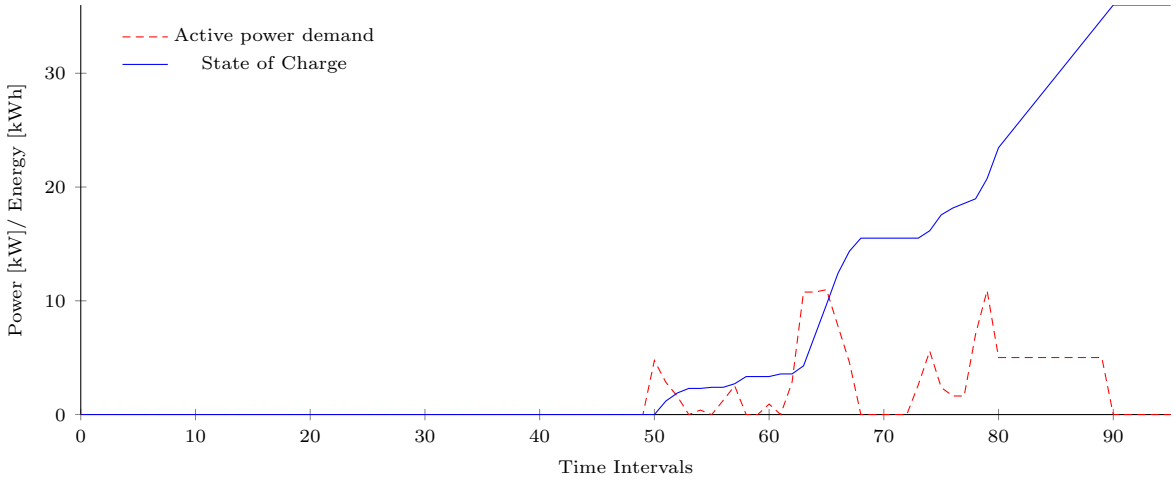

Figure 4.9: Active power demand and state of charge of EV 2

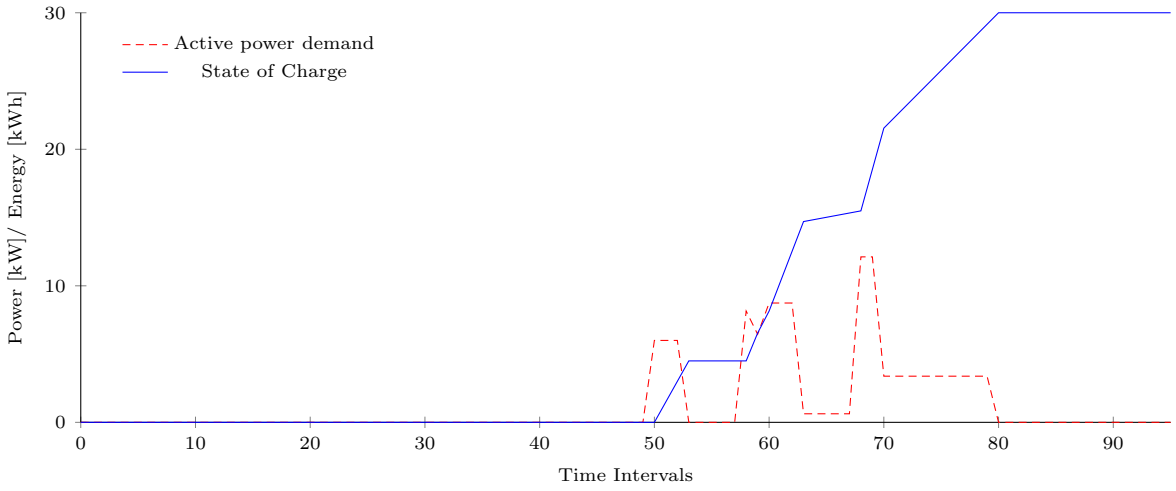

Figure 4.10: Active power demand and state of charge of EV 3

(with and without deadline extensions) is performed. This new plan, according to Algorithm 3.2, tries to compensate for the next 5 intervals ($K = 5$) the deviation caused by wrong prediction of uncontrollable loads. As a result, after 5 intervals, Equation 3.1 is violated again at *interval*[58]. It is then decided that the length of the deviation of uncontrollable loads is longer than expected (K). Thus the 2^{nd} replanning with the same strategy as the 1^{st} one is performed. For the same reason, the 3^{rd} and 4^{th} replannings are done at *interval*[63] and *interval*[68] respectively, i.e. every 5 intervals after the last replan. These replan times can also be found in Figure 4.11.

Then at *interval*[69] and *interval*[70], because of the sudden increase of uncontrollable loads, replanning is also preformed. After that the curve of accumulated replan times stays unchanged.

In Figure 4.12, it is shown that compared with taking no actions (blue dashed line), the new plan (black solid line) stays more close to the original plan (red dotted line). It can be calculated that the new planning can improve the proximity factor Q' by 14.1% ((19844.22 − 17046.94)/19844.22). However this is at the cost of delaying the deadlines of EV 2 and EV 3 by 10 intervals therefore penalties for these must be

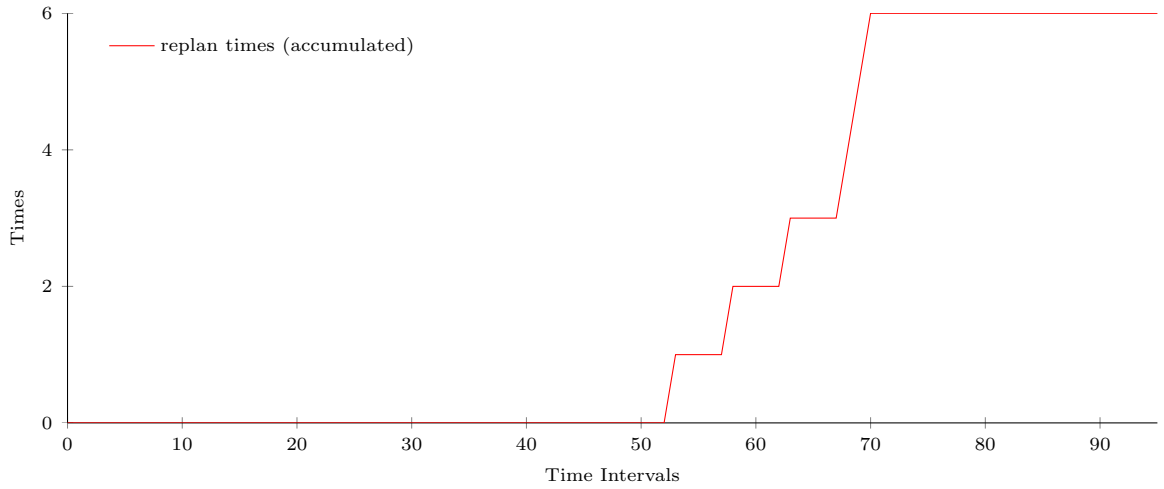

Figure 4.11: How many times replan has taken place (accumulated)

taken into account. Using Equation 4.4, it can be calculated that with real-time control, $\hat{Q}'$ can be improved by 9.1% $((19844.22 - (17046.94 + 50 * 10 * 2))/19844.22)$ compared with taking no actions. While the optimal solution obtained using method depicted in Figure 4.2 gives 33.3% improvement $((19844.22 - 13232.03)/19844.22)$. No obvious oscillation can be observed in Figure 4.12.

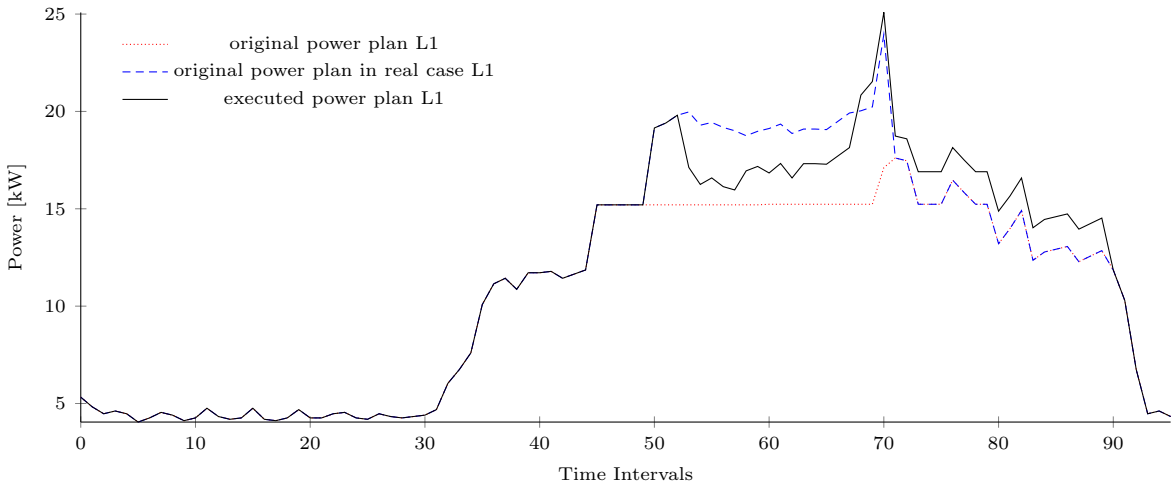

Figure 4.12: Original plan (predicted and real life) and final executed plan of phase 1

$\gamma = 300$
In this experiment, since the penalty of charging deadline extensions is high, during the experiment, only the deadline of EV 2 is extended by 10 to $interval[90]$, as shown in Figure 4.13. Except for this factor, the process of this experiment is basically the same as that of the 1^{st} one.

In Figure 4.14, it is shown that compared with taking no actions (blue dashed line), the new plan (black solid line) stays more close to the original plan (red dotted line). It can be calculated that the new planning can improve the proximity factor Q' by 12.7% $((19844.22 - 17317.18)/19844.22)$. Using Equation 4.4 however, it can be calculated that with real-time control, $\hat{Q}'$ decreases by 2.4%

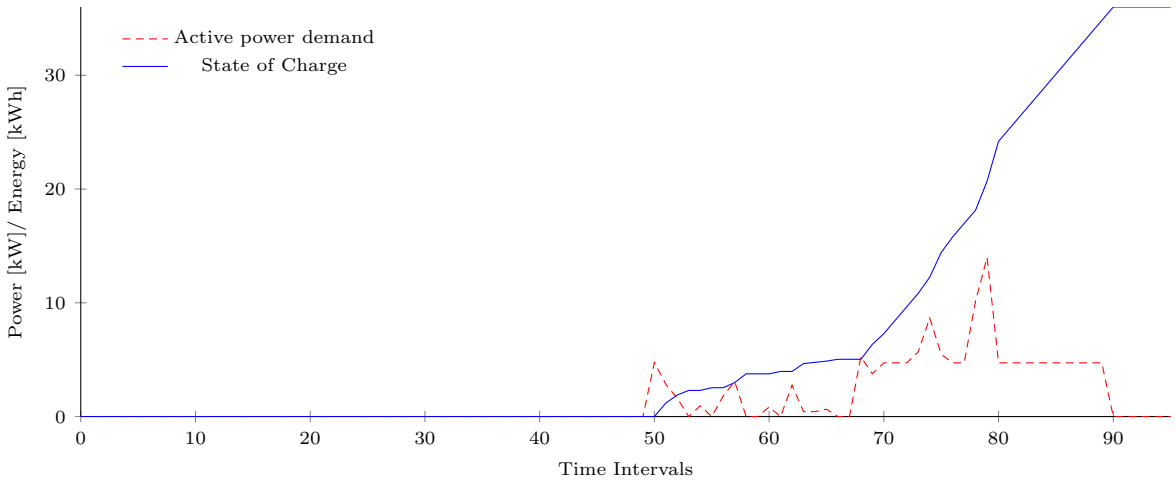

Figure 4.13: Active power demand and state of charge of EV 2

$((19844.22 - (17317.18 + 300 * 10))/19844.22)$ compared with taking no actions. In contrast to the 9.1% improvement when $\gamma = 50$, in this experiment, the real-time control algorithm fails at its task. The reason is that when punishment for EV charging deadline extension is high, its must be guaranteed that few oscillations happen. When K is not set properly, high penalty may lead to poor performance of Algorithm 3.2. But compared with the last experiment, EV 3 can finish charging on time. No obvious oscillation can be observed in Figure 4.14. Overloading test is not conducted in this case either.

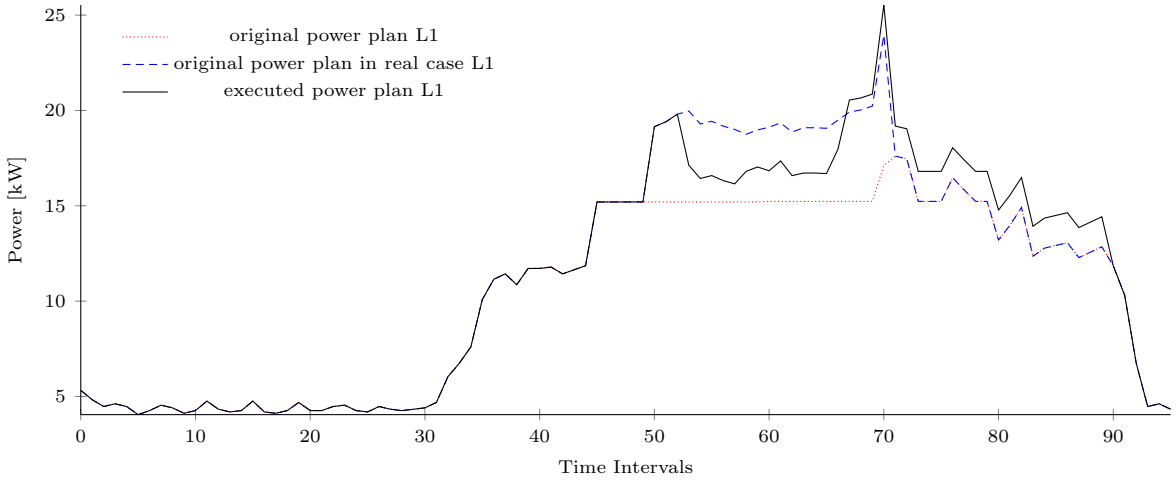

Figure 4.14: Original plan (predicted and real life) and final executed plan of phase 1

4.2.3 Case 3: EV leaves earlier than predicted

The detailed settings of the EV charging jobs in this case can be found in Table 4.3. As can be observed in it, EV 1 leaves at $interval[55]$ instead of at $interval[60]$ as expected. As a result, the real-time control algorithm actuates when EV 1 leaves (Figure 4.16), trying to compensate its charging power with other available charging jobs using Algorithm 3.2. In Figure 4.15, it is shown that compared with taking no actions (blue dashed line), the new plan (black solid line) stays more close to the original plan (red dotted line). Using Equation 4.4, it can be calculated that the new planning can improve the proximity factor Q'

	(Predicted) Start time	End time	Predicted end time	Max charging power[kW]	Capacity and Setpoint[kWh]
EV 1	45	55	60	17.1	30
EV 2	50	80	80	14.1	36
EV 3	50	70	70	17.1	30

Table 4.3: Detailed information of all EV charging jobs

by 49.7% ((3907.91 − 1967.41)/3907.91). This result is satisfactory even compared with the improvement given by the optimal solution, 65.1% ((3907.91 − 1364.70)/3907.91). After *interval*[55], EV 1 leaves without being fully charged (Figure 4.17) and no oscillation is observed in Figure 4.16. All other EVs are charged on time and no overloading test is done in this case.

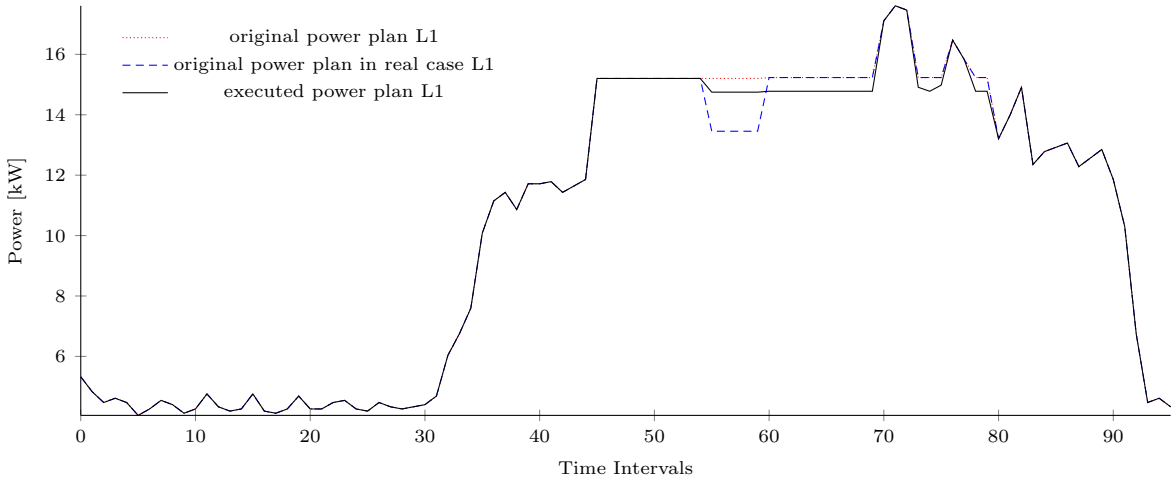

Figure 4.15: Original plan (predicted and real life) and final executed plan of phase 1

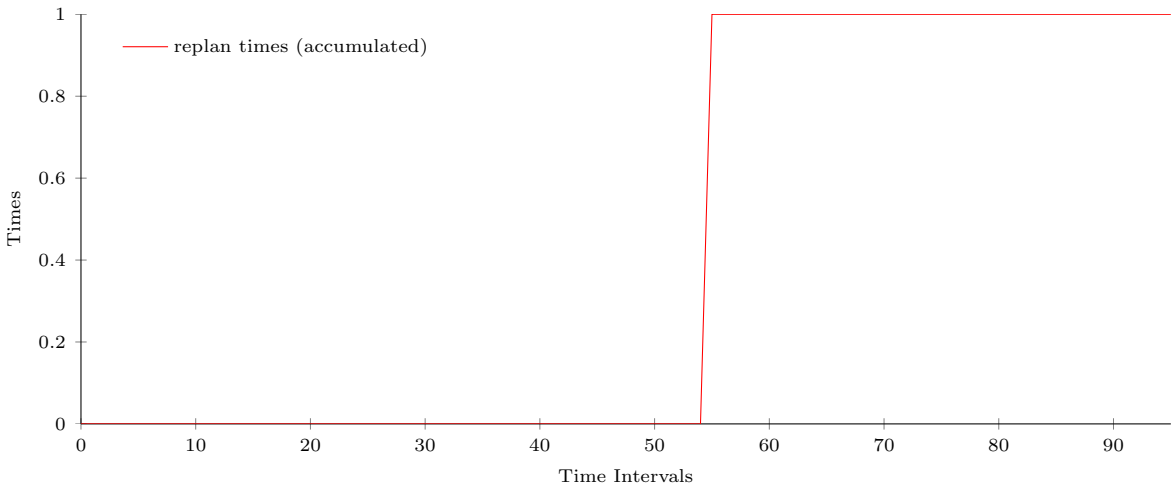

Figure 4.16: How many times replan has taken place (accumulated)

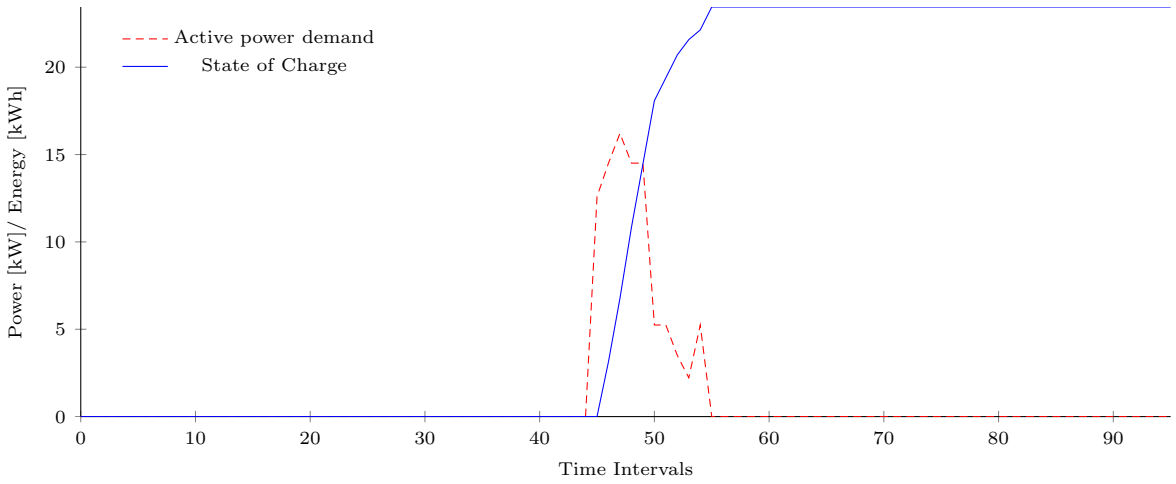

Figure 4.17: Active power demand and state of charge of EV 1

4.2.4 Case 4: EV arrives later than predicted

The detailed settings of the EV charging jobs in this case can be found in Table 4.4. As can be observed in it, EV 3 arrives at *interval*[55] instead of at *interval*[50] as expected. As a result, the real-time control algorithm first actuates at the predicted start time of EV 3 (*interval*[50]) as can be seen in Figure 4.19. This is similar to what happens when EV leaves earlier than predicted, the master controller tries to compensate the "missing" charging power with other available charging jobs at the time. Later when EV 3 actually arrives at *interval*[55], one more charging job becomes available and replanning is performed again (Figure 4.19). This ensures that all EVs can be fully charged on time.

	Start time	Predicted start time	(Predicted) End time	Max charging power[kW]	Capacity and Setpoint[kWh]
EV 1	45	45	60	17.1	30
EV 2	50	50	80	14.1	36
EV 3	55	50	70	17.1	30

Table 4.4: Detailed information of all EV charging jobs

In Figure 4.18, it is shown that compared with taking no actions (blue dashed line), the new plan (black solid line) stays more close to the original plan (red dotted line). This can be proved using Equation 4.4. It can be calculated that the new planning can improve the proximity factor Q' by 5.1% $((4472.14 - 4243.28)/4472.14)$. The improvement is not as obvious as the last case compared with the improvement achieved by the optimal solution, 100% (the integrals of new plan and original one in Figure 4.18 are the same). However this is due to the fact that the total energy consumption of the new plan is higher than that of taking no actions. The difference can also be observed in Figure 4.18 since the integral of the new plan (black solid line) is greater than that of taking no actions (blue dashed line). As can be seen in Figure 4.20, this ensures that EV 3 is fully charged on time. After *interval*[55], no more replanning is performed. In general, no oscillation is observed in Figure 4.19. Additionally, all EVs are charged on time and no overloading test is conducted in this case.

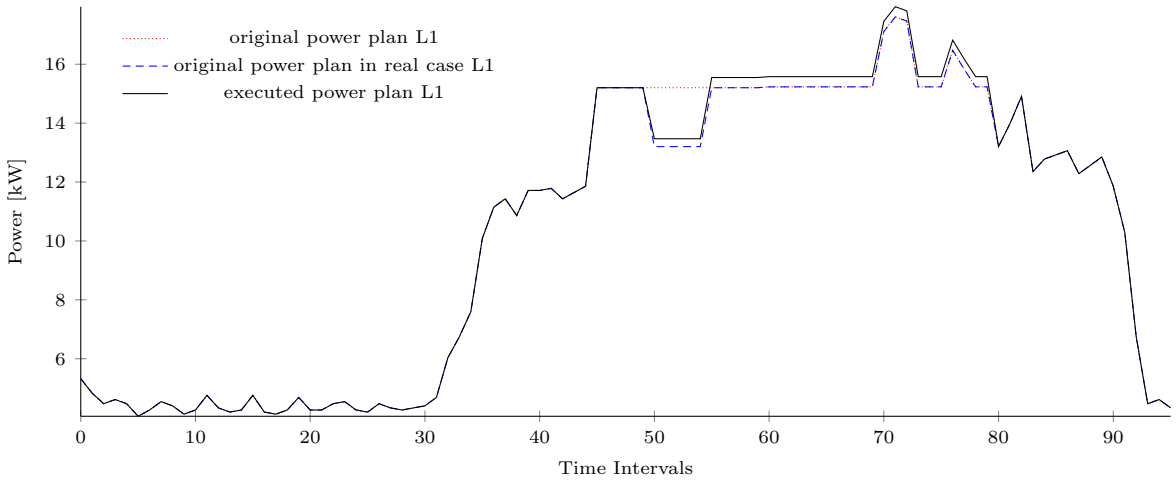

Figure 4.18: Original plan (predicted and real life) and final executed plan of phase 1

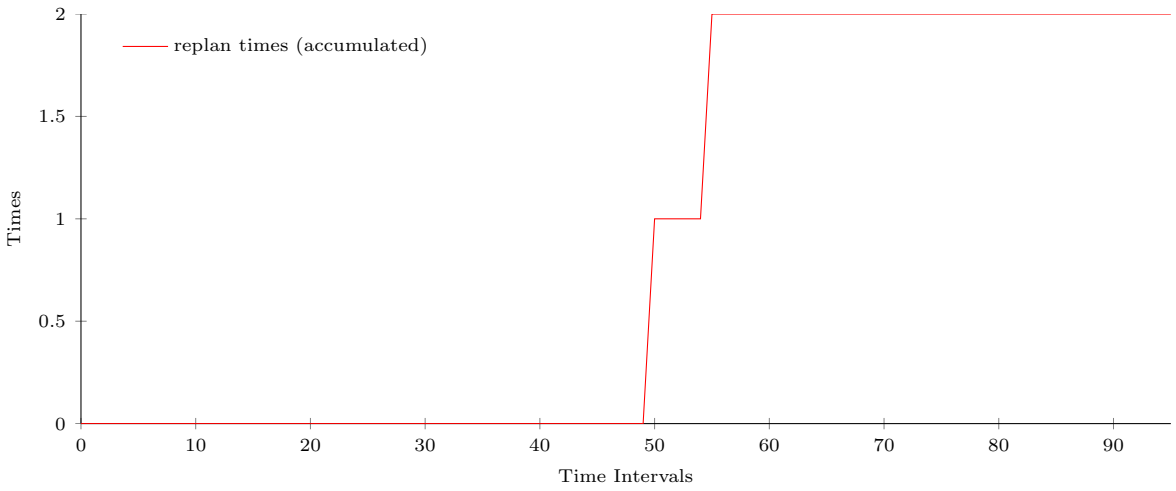

Figure 4.19: How many times replan has taken place (accumulated)

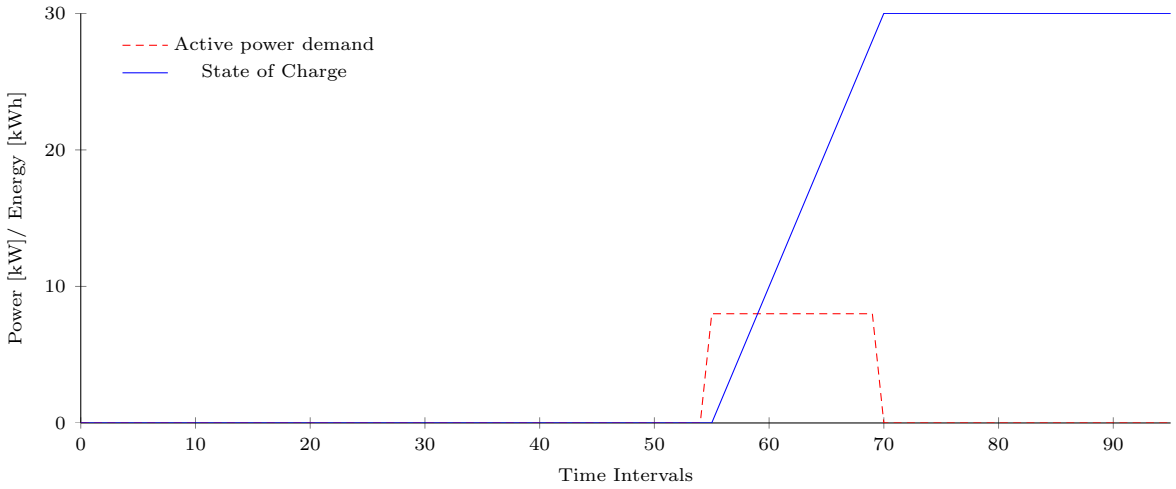

Figure 4.20: Active power demand and state of charge of EV 3

4.2.5 Case 5: Uncontrollable load shifts over time

	(Predicted) Start time	(Predicted) End time	Max charging power[kW]	Capacity and Setpoint[kWh]
EV 1	45	60	17.1	30
EV 2	50	80	14.1	36
EV 3	50	70	17.1	30

Table 4.5: Detailed information of all EV charging jobs

The detailed settings of EV charging jobs in this case can be found in Table 4.5. In order to create an unexpected uncontrollable power peak, from time *interval*[45] to *interval*[54], the real uncontrollable load is 1.4 times the predicted uncontrollable load in all three phases. Also to create an unexpected power drop, from time *interval*[60] to *interval*[69], the real uncontrollable load is half of the predicted uncontrollable load in all three phases. Combined together they create the effect of shifting uncontrollable load over time. This effect is shown in Figure 4.21.

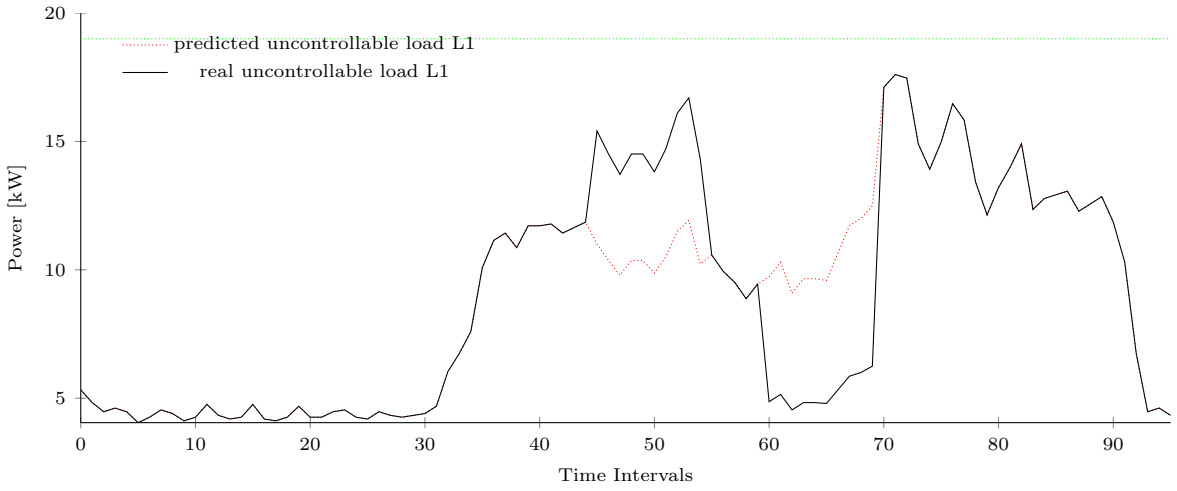

Figure 4.21: Predicted uncontrollable load and real uncontrollable load of phase 1

As shown in Figure 4.21, the integral of the predicted uncontrollable profile (red dotted line) in phase L1 is almost the same as that of the real uncontrollable profile (black solid line). The predicted energy consumption of uncontrollable load over the day is $215.414kWh$, slightly more than $212.899kWh$ in real case. In this case, two experiments are conducted, with and without overloading control. In the second experiment (with overloading control), the maximum overall power allowed is set to $19000kW$, as can be seen as the green dotted line in Figure 4.21.

Without overloading control

In this experiment, before *interval*[48], the real-time control algorithm never performs replanning. At *interval*[48], due to the changes in Figure 4.21, Equation 3.2 stands ($M = 4$). Therefore the 1^{st} replanning (with and without deadline extensions) is performed. This new plan, according to Algorithm 3.2, tries to compensate for the next 5 intervals ($K = 5$) the deviation caused by wrong prediction of uncontrollable loads. As a result, after 5 intervals, Equation 3.1 is violated again at *interval*[53]. It is then decided that the length of the deviation of uncontrollable loads is longer than expected (K). Thus the 2^{nd} replanning with the same strategy as the 1^{st} one is performed. The deviation at *interval*[53] will be compensated for

the following 5 intervals. However since in this case the power peak will only last for two more intervals (until *interval*[54]), overcompensation can be observed from *interval*[55] *to interval*[57] in Figure 4.22.

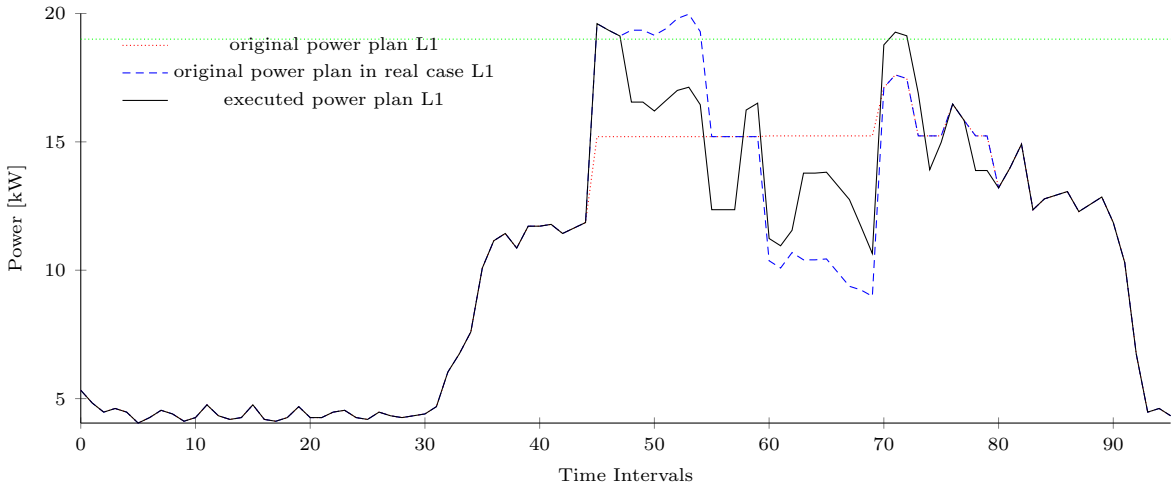

Figure 4.22: Original plan (predicted and real life) and final executed plan of phase 1

Then at *interval*[63], due to the changes in Figure 4.21 from *interval*[60] to *interval*[63], Equation 3.2 stands again. Therefore the 3^{rd} replanning is performed. This new plan, according to Algorithm 3.2, tries to compensate for the next 5 intervals ($K = 5$) the deviation caused by wrong prediction of uncontrollable loads. As a result, after 5 intervals, Equation 3.1 is violated again at *interval*[68]. It is then decided that the length of the deviation of uncontrollable loads is longer than expected (K). Thus the 4^{th} replanning with the same strategy as the 3^{rd} one is performed.

At *interval*[69], replanning is also performed according to Figure 4.23. The reason is that at the beginning of *interval*[69], EV 3 is almost fully charged (EV 1 has finished charging) so controllable power at disposal is limited. Furthermore, Algorithm 3.2 tries to spread this power over the next 5 intervals to compensate deviation. Therefore even though at *interval*[69] uncontrollable power rises compared to that at *interval*[68], replanning is still triggered.

In Figure 4.22, it is shown that compared with taking no actions (blue dashed line), the new plan (black solid line) stays more close to the original plan (red dotted line). This can be proved using Equation 4.4. It can be calculated that the new planning can improve the proximity factor $\hat{Q}'$ by 32.6% $((21408.07 - 14419.96)/21408.07)$. While the optimal solution obtained using method depicted in Figure 4.2 gives 93.4% improvement $((21408.07 - 1408.68)/21408.07)$. At the same time, all EVs are charged on time. No obvious oscillation can be observed in Figure 4.22.

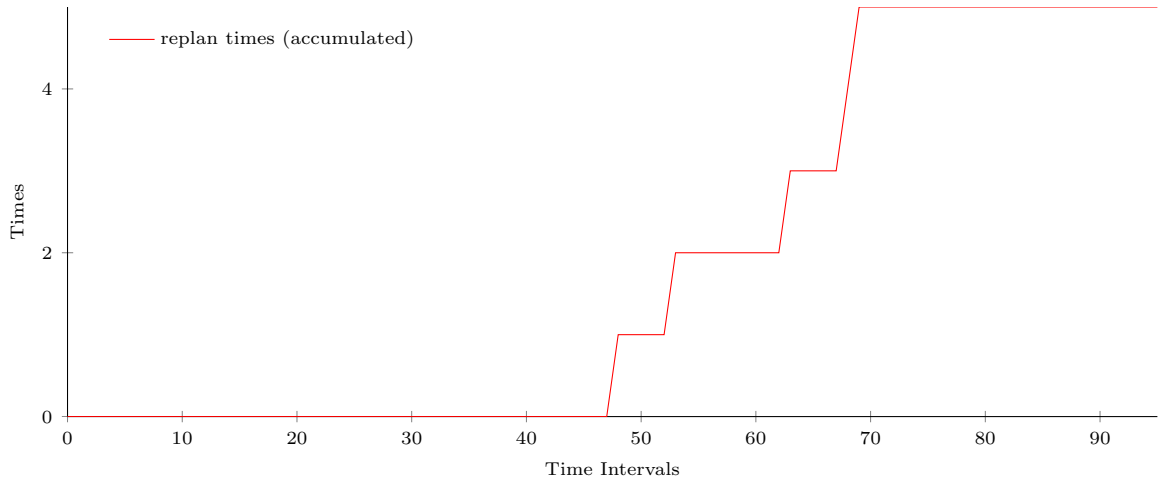

Figure 4.23: How many times replan has taken place (accumulated)

With overloading control

In this experiment, before *interval*[45], the real-time control algorithm never performs replanning. Since $P_{total}(45)$ is greater than the maximum overall power allowed (green dotted line in Figure 4.24), at *interval*[45], replan is performed as shown in Figure 4.25. As a result, at *interval*[46] and *interval*[47], P_{total} stays within bounds. Then at *interval*[48], Equation 3.2 stands ($M = 4$). Therefore the 2^{nd} replanning (with and without deadline extensions) is performed. This new plan, according to Algorithm 3.2, tries to compensate for the next 5 intervals ($K = 5$) the deviation caused by wrong prediction of uncontrollable loads. As a result, after 5 intervals, Equation 3.1 is violated again at *interval*[53]. It is then decided that the length of the deviation of uncontrollable loads is longer than expected (K). Thus the 3^{rd} replanning with the same strategy as the 2^{nd} one is performed. The deviation at *interval*[53] will be compensated for the following 5 intervals. However since in this case the power peak will only last for two more intervals (until *interval*[54]), overcompensation can be observed from *interval*[55] *to interval*[57] in Figure 4.24.

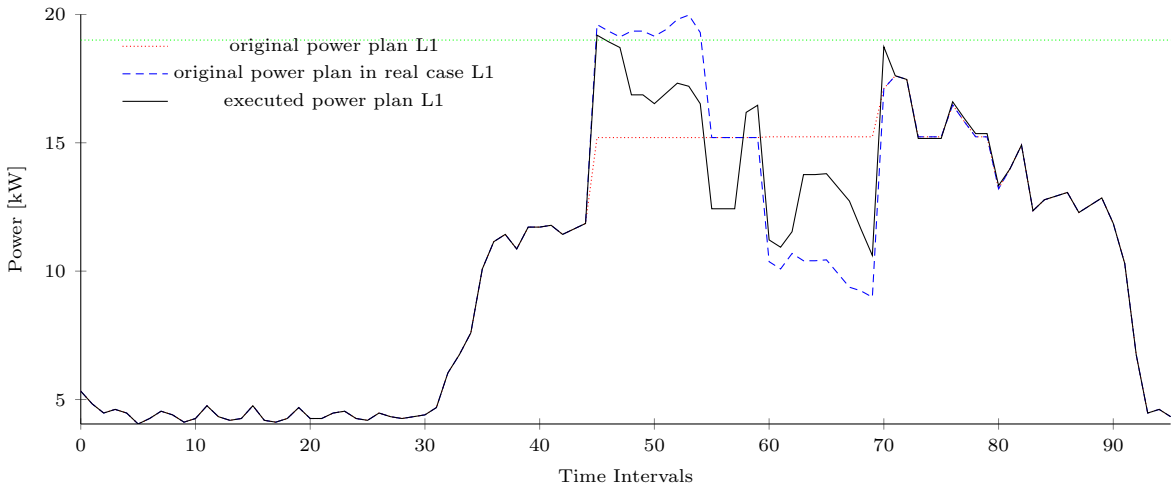

Figure 4.24: Original plan (predicted and real life) and final executed plan of phase 1

Then at *interval*[63], due to the changes in Figure 4.21, Equation 3.2 stands again. Therefore the 4^{th} replanning is performed. This new plan, according to Algorithm 3.2, tries to compensate for the next 5 intervals ($K = 5$) the deviation caused by wrong prediction of uncontrollable loads. As a result, after 5 intervals, Equation 3.1 is violated again at *interval*[68]. It is then decided that the length of the deviation of uncontrollable loads is longer than expected (K). Thus the 5^{th} replanning with the same strategy as the 4^{th} one is performed.

At *interval*[69], replanning is also performed according to Figure 4.25. The reason is the same as that of the first experiment. As a result, overcompensation can be observed at *interval*[70] in Figure 4.24. Later at the beginning of *intetval*[71], $P_{total}(71)$ is greater than the maximum overall power allowed (green dotted line in Figure 4.24), thus replan is performed as shown in Figure 4.25. As a result, at *interval*[71] and *interval*[72], P_{total} stays within bounds.

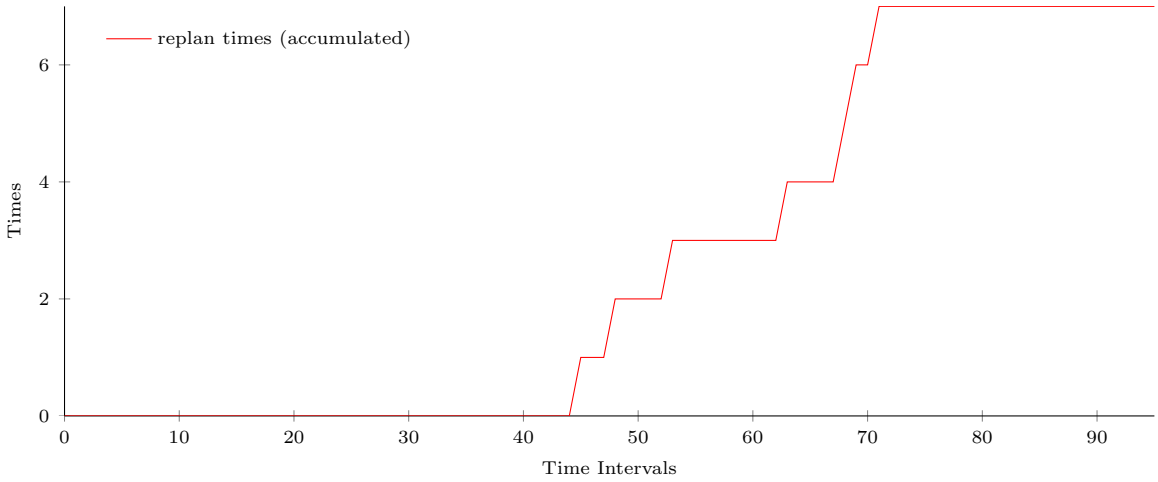

Figure 4.25: How many times replan has taken place (accumulated)

In Figure 4.24, it is shown that compared with taking no actions (blue dashed line), the new plan (black solid line) stays more close to the original plan (red dotted line). This can be proved using Equation 4.4. It can be calculated that the new planning can improve the proximity factor $\hat{Q}'$ by 35.6% $((21408.07 - 13783.21)/21408.07)$. At the same time, all EVs are charged on time. No obvious oscillation can be observed in Figure 4.22. Moreover, compared with the first experiment where overloading control is not introduced, upon detection of overloading problem, it is quickly handled. (Figure 4.22 and Figure 4.24)

4.3 Conclusion

Based on analysis mentioned before in this chapter, the performance of the real-time control algorithm can be summarized in Table 4.6. As can be seen in Table 4.6, except for Case 1 and the second experiment in Case 2 ($\gamma = 300$), Algorithm 3.2 improves the proximity factor (Q' or $\hat{Q}'$ with deadline extension). In Case 3, Case 4 and Case 5, compared with taking no actions, the resulted new plan of the real-time control algorithm remians closer to the original plan without interfering EV charging deadlines. In the first experiment of Case 2 ($\gamma = 50$), the proximity factor $\hat{Q}'$ is also improved at the cost that the finishing time of two EV charging jobs are postponed. Compared with the second experiment, it is shown that by extending the EV charging deadlines of certain job(s), the proximity factor $\hat{Q}'$ can be further minimized. No obvious oscillation can be observed in those cases either.

Case NO.	Improvement	Optimal improvement
Case 1	2.7%	33.3%
Case 2 ($\gamma = 50$)	9.1%	33.3%
Case 2 ($\gamma = 300$)	-2.4%	33.3%
Case 3	49.7%	65.1%
Case 4	5.1%	100%
Case 5 (no overloading control)	32.6%	93.4%
Case 5 (with overloading control)	35.6%	93.4%

Table 4.6: Comparison of improvement achieved by Algorithm 3.2 and optimal improvement

However, in Case 1, because of the exhaustion of potential of compensating the drop of uncontrollable loads during a short time period (5 intervals), oscillation is present twice. The real-time control algorithm can still improve the proximity factor Q' a bit though. In the second experiment of Case 2, due to the high punishment for EV charging deadline extensions and improper selection of K, Algorithm 3.2 fails at its task.

In more realistic scenarios where the optimal improvement is higher than 50% (Case 3, Case 4 and Case 5), Algorithm 3.2 tends to behave as expected and the improvement is generally promising. Although in Case 4 the actual improvement (5.1%) is not that obvious compared with the optimal solution (100%), the shape of the executed profile shown as the black solid line in Figure 4.18 is as predicted. By tuning coefficients such as the degree at which P_{dev} will be corrected (β), it is possible to gain a better improvement.

In conclusion, Algorithm 3.2 can generally fulfill its duty of forcing the executed overall power plan stay close to the original one. In some experimental cases (e.g. Case 2), this might be accomplished at the cost of extending deadlines of EV charging jobs. Taking the penalty of the extensions into consideration, in certain cases (e.g. $\gamma = 300$ in Case 2), the real-time control algorithm may fail at this task. While in realistic cases, Algorithm 3.2 tends to offer satisfactory improvement even compared with what is gained by the optimal solution. Note that the optimal solution is obtained under the assumption that both the shape of uncontrollable loads and EV charging deadlines are not considered. Occasionally, oscillations can be observed (e.g. Case 1).

In Case 5, it is proven upon detection, overloading problem is solved as soon as possible by Algorithm 3.2. The introduction of overloading control may even improve the proximity factor $\hat{Q}'$.

Algorithm 3.2 is by design device agnostic since it is based on Algorithm 2.1 which is proven to be so. Moreover, deadline extension can be applied not only to buffer time shiftable devices (but to time shiftable devices).

5 Conclusion and Future Work

This master thesis introduces the first attempts to implement a real-time control algorithm in Triana on SASensor Open Platform. Based on the planning algorithms presented in [5] and [24], the proposed algorithm tries to meet the following requirements,

- Upon detection, it must be ensured that overloading problem will be solved as soon as possible.
- Final overall power profile remains as close to its original plan as possible.
- Charging deadlines of EVs set by end users are not violated if preferred.

By exploring possible reasons of deviation from original plan, four base cases are determined as follows.

- Case 1: uncontrollable load is lower than predicted
- Case 2: uncontrollable load is higher than predicted
- Case 3: EV leaves earlier than predicted
- Case 4: EV arrives later than predicted

For each of them a clear strategy is decided in section 3.1.2. Finally the algorithm is tested in simulation environment with Triana network model of Lochem.

As discussed in section 4.3, Algorithm 3.2 can generally fulfill its duty of forcing the executed overall power plan stay close to the original one (Table 4.6). In some experimental cases (e.g. Case 2), this might be accomplished at the cost of extending deadlines of EV charging jobs. Taking the penalty of the extensions into consideration, in certain cases (e.g. $\gamma = 300$ in Case 2), the real-time control algorithm may fail at this task. While in realistic cases (Case 3, Case 4 and Case 5), Algorithm 3.2 tends to offer satisfactory improvement even compared with what is gained by the optimal solution.

There is still room left to improve the performance of Algorithm 3.2 though. First of all, coefficients and parameters can be tuned such as the minimal length of consecutive intolerable P_{dev} (i.e. M), the predicted length of drop/peak of $P_{no\_control}$ (i.e. K) and the degree at which P_{dev} will be corrected (i.e. β). Additionally a probabilistic prediction of the pattern of uncontrollable load can certainly help to make informed decisions.

This thesis mainly answers two research questions, namely,

- How to design the real-time control algorithm incorporated into the Triana simulator to realize the aforementioned objective of the system?

  To answer the first research question, different strategies are explored in section 3.1.1. The real-time control algorithm is designed to be time-based and it will perform a replan to approximate the original profile. Once replan is necessary, the designed real-time control algorithm (Algorithm 3.2) tries to adapt original power plan with the deviation measured at the time. Taking the (adjusted) original overall plan as the desired profile while replanning, this time-based algorithm ensures that the final executed plan approximates the original plan (sometimes at the cost of charging deadline extensions). In simulation it also reacts to overloading problem quickly.

- What is the best design for the API of the Triana simulator to access RTD?

A plan to design the API of the Triana to access RTD is proposed to answer the second research question. Collective active power per phase is determined to be the necessary RTD for the field test in section 3.1.3. All EV charging poles should be metered as well. To access those RTD, polling is decided a sufficient choice in section 3.1.4. With the design of the API, it is possible to apply the real-time control algorithm into field test in the future.

5.1 Future work

Future work may include taking cases when EV(s) arrive earlier than expected or EV(s) leave later than expected into account. Although those two events will not cause deviation from original plan, they can provide further flexibility like Case 1 when uncontrollable load is lower than expected. Another possible case would be when real state of charge (SOC) or setpoint of EV(s) differs from prediction.

It would also be possible to apply this real-time control algorithm into a more complex case with multiple layers of group controllers, namely house controller layer and master controller layer. Another direction would be to include other kinds of controllable devices into the use case. Theoretically, Algorithm 3.2 is device agnostic because it is based on the planning algorithm 2.1 where in each iteration group controllers check the flexibility of each of its child controllers without asking which kind of controllers they are. Additionally, deadline extension can be applied to buffer time shiftable devices as well as time shiftable devices.

Last but not the least, like mentioned before, it would be preferable to perform a field test of the real-time control algorithm. However, issues such as communication time between controllers are still yet to be solved. Another difficulty might be dealing with the noise of power consumed by LV cables and the system error introduced by load-flow simulation.

Acronyms

μCHP	micro-Combined Heat and Power
API	Application Programming Interface
BAR	base address register
BIM	Break Interface Module
CAES	Computer Architecture and Embedded Systems group
CCU	Central Control Unit
CIM	Current Interface Module
DG	Distributed Generation
DMA	direct memory access
DS	Distributed Storage
DSM	Demand Side Management
EV	electric vehicle
GUI	graphical user interface
HMV	High Medium Voltage
HV	high voltage
ILP	Integer Linear Problem
IM	Interface Module
IPC	Inter-process Communication
KCL	Kirchhoff's Current Law
KVL	Kirchhoff's Voltage Law
KVM	Kernel-based Virtual Machine
LV	low voltage
MLV	Medium Voltage
MPC	Model Predictive Control
MV	medium voltage

OS operating system

PC personal computer

PCI Peripheral Component Interface

PHEV plug-in hybrid Electric Vehicle

PV Photovoltaics

RMS Root Mean Square

RTD real-time measurement data

SMS Shared-Memory Server

SOC state of charge

VCU Versatile Communication Unit

VIM Voltage Interface Module

VM virtual machine

VMM Virtual Machine Monitor

References

[1] Anoop Ambooken. *A graphical framework for the SASensor*. University of Twente, 2013.

[2] V Bakker. *Triana: a control strategy for Smart Grids*. PhD thesis, University of Twente, 2011.

[3] Jianhua Che, Qinming He, Qinghua Gao, and Dawei Huang. Performance measuring and comparing of virtual machine monitors. In *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, volume 2, pages 381–386, Dec 2008.

[4] Clark W Gellings. The concept of demand-side management for electric utilities. *Proceedings of the IEEE*, 73(10):1468–1470, 1985.

[5] Marco ET Gerards, Hermen A Toersche, Gerwin Hoogsteen, Thijs van der Klauw, Johann L Hurink, and Gerard JM Smit. Demand side management using profile steering. 2015.

[6] Yasunori Goto. Kernel-based virtual machine technology. *Fujitsu Scientific and Technical Journal*, 47:362–368, 2011.

[7] Timo Hirt. Kvm-the kernel-based virtual machine. *Red Hat Inc*, 2010.

[8] G Hoogsteen. Impact of peak electricity demand in distribution grids: A stress test. 2015.

[9] Gerwin Hoogsteen. *Simulating the effects of smart grid technologies on power quality*. University of Twente, 2013.

[10] Gerwin Hoogsteen, Albert Molderink, Johann L Hurink, and Gerard JM Smit. Managing energy in time and space in smart grids using triana. In *Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2014 IEEE PES*, pages 1–6. IEEE, 2014.

[11] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.

[12] A Cameron Macdonell. Shared-memory optimizations for virtual machines. Master's thesis, University of Alberta, 2011.

[13] Omar Mansour. *Open Platform Technical Report*. Locamation B.V., 2014.

[14] Albert Molderink. *On the three-step control methodology for Smart Grids*. University of Twente, 2011.

[15] Alexandre Oudalov, Rachid Cherkaoui, and Antoine Beguin. Sizing and optimal operation of battery energy storage system for peak shaving application. In *Power Tech, 2007 IEEE Lausanne*, pages 621–625. IEEE, 2007.

[16] Beata Polgari. *First steps to make SASensor an Open Platform system*. Delft University of Technology, 2011.

[17] Qumranet. White paper: Kvm kernel-based virtualization driver, 2006. [Online].

[18] J Scott, P Vaessen, and F Verheij. Reflections on smart grids for the future. *Dutch Ministry of Economic Affairs*, pages 1–10, 2008.

[19] Abraham Silberschatz, Peter B Galvin, and Greg Gagne. *Operating system concepts*, volume 8. Wiley, 2013.

[20] James Stewart. Calculus early transcendentals, 6e. *Belmont, CA: Thompson Brooks/Cole*, 2006.

[21] Hermen A Toersche, Vincent Bakker, Albert Molderink, S Nykamp, Johann L Hurink, and Gerard JM Smit. Controlling the heating mode of heat pumps with the triana three step methodology. In *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*, pages 1–7. IEEE, 2012.

[22] Antonio Trias. The holomorphic embedding load flow method. In *Power and Energy Society General Meeting, 2012 IEEE*, pages 1–8. IEEE, 2012.

[23] B Tubben. Programmaplan in4energy. April, 2012.

[24] Thijs van der Klauw, Marco ET Gerards, Gerard JM Smit, and Johann L Hurink. Optimal scheduling of electrical vehicle charging under two types of steering signals. In *Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2014 IEEE PES*, pages 1–6. IEEE, 2014.

[25] Wikipedia. Polling (computer science) — wikipedia, the free encyclopedia, 2015. [Online; accessed 13-March-2015].

[26] Ray Daniel Zimmerman. *Comprehensive distribution power flow: modeling, formulation, solution algorithms and analysis*. PhD thesis, Cornell University, 1995.