Westfälische
Wilhelms-Universität
Münster

Master thesis

Energy storage in smart homes: grid-convenience versus self-use

by Jannik Hüls

Westfälische Wilhelms-Universität Münster
Fachbereich Mathematik und Informatik
Safety-critical systems group

supervised by
Prof. Dr. Anne Remke
Dr. Henrik Blunck

March 2016

wissen.leben
WWU Münster

Westfälische
Wilhelms-Universität
Münster

Masterarbeit

Energiespeicherung in smart homes: Netzverträglichkeit versus Eigenverbrauch

von Jannik Hüls

Westfälische Wilhelms-Universität Münster
Fachbereich Mathematik und Informatik
Arbeitsgruppe Sicherheitskritische Systeme

betreut durch
Prof. Dr. Anne Remke
Dr. Henrik Blunck

März 2016

wissen.leben
WWU Münster

Acknowledgements

I would like to thank Anne Remke for her detailed advices, constructive feedback and very encouraging communication. It was nice to recognize that you were interested in my work.

Furthermore I would like to thank Hamed Ghasemieh for his detailed introduction into the used tools code and for taking time to answer my many questions.

In addition I would like to thank my parents for their support during my entire studies and my girlfriend for her listening skills whenever things went wrong.

Abstract

The number of local power generation units, such as photovoltaic panels (PV), increased enormously in recent years. Their production depends on the current weather and thus is highly variable. This fluctuation in production means a major challenge towards the stability of the power grid. The use of local energy storages may help to ensure that the locally generated power is fed into the grid in a *grid-convenient* way. It may also support clients to increase the *self-use* of local generated power and to increase the *survivability* of their homes in case of power outages.

In the first part of this thesis we compare the interest of the power operator, i.e. grid-convenience with the interest of the client, i.e. self-use and survivability for four different battery management strategies i) direct loading, ii) delayed loading, iii) peak shaving and iv) prediction based loading. We use Hybrid Petri Nets with a single general one-shot transition (HPnG) to model a smart home with local power generation, local energy storage and different battery management strategies. Recent algorithms for model checking HPnGs are used to assess the above mentioned measures of interest. Whenever good predictions of production and demand exist, we are able to show that grid-convenience does not decrease the self-use nor the survivability.

Greater local energy storages increase the proportion of self-use. Today electric vehicles (EVs) offer very high battery capacities. Within the second part of this thesis we will assess the grid-convenient charging of EVs. Whenever the available time for charging EVs is known, the charging can be delayed to charge the EV *just-in-time*. This offers battery capacities to the smart grid for a coordinated charging. The grid then is able to decide when to charge the car, i.e. preferably in times the grid is not heavily used. As a measure of grid-convenience we will assess the average available battery capacity. The car has to be preferably high charged when the client returns. Thus the *integrity* of the charging process indicates the clients interest.

Again we will compare the interest of the power operator, i.e. a preferably high average available battery capacity with the integrity of the charging process, which is the clients interest. The HPnG model covers the charging of EVs separated from a smart house since it is a stand-alone process. We are able to show that a just-in-time charging is suitable to both provide a satisfying integrity and high average battery capacity for a coordinated use. Indeed the client has to actively determine the charging time and thus consciously has to have the will to support grid-convenience.

List of Figures

Table of Contents

1 Introduction

With the still ongoing energy revolution the use of renewable energy highly increased in the recent years. As shown in [1] today at least 30% of the generated energy is produced by renewable sources. Until year 2050 this number is expected to increase up to 80%. Even though this is a huge effort towards environmental friendly energy generation, the growing significance of renewable energy has its drawback, as well. By the use of photovoltaic panels or wind turbines there can occur strong gradients in the amount of produced energy caused by weather changes. It is a serious task to the grid to handle this alternating production and to stay stable. Related work states that due to the decentralized renewable power generation the number of grid faults is expected to increase up to 60% until 2020. [4] When managed in a grid-convenient way, local energy storage may lead to a more stable grid. We assess several battery management strategies for smart homes and compare their *grid-convenience*, *self-use* and *survivability* values.

The highest possible self-use may be reached using great local storages. But this additional battery capacity may also be used grid-conveniently. It can be offered to the grid for load balancing and frequency compensation. Greater available battery capacaties can be achieved by buying greater batteries, but also are offered by electric vehicles (EVs). Due to the rising significance of EVs, we cover them as the source of high battery capacity. We focus on the charging process of EVs and investigate a charging strategy that both may act grid-conveniently and satisfies the clients interests, i.e. the *integrity* of the charging process.

1.1 Motivation

We use Hybrid Petri Nets with a single general one-shot transitions (HPnGs) [5] to model smart homes that consist of a local power production, demand and battery storage. The single one-shot transition is used to model the randomly distributed length of a power outage, during which the local storage or production is used to fulfill the demand. The model consists of two power sources, i.e. the local production and the grid. We extend the model presented in [3] to explicitly incorporate the battery and be able to approximate losses that occur due to charging and discharging the battery. We do not consider battery aging or any non-linear effects that occur in the battery when the battery is almost full or empty.

Likewise HPnGs are used to model the charging of electric vehicles. We do not simply extend the smart home by a greater battery, but assess an advanced model since we want to use the general transition in another purpose. The charging of the electric vehicle is started by determining the total time the car is available for charging. We then are able to model the clients recurrence by a general transition. We have one power source, which both can be the grid or a local power production. This model can be seen as a separate extension of the first presented smart home model.

Recent model checking algorithms [6] are used to investigate the measures of interest. We are able to assess the probability that the battery is able to power the house in case of a grid outage. Alike model checking is used to investigate the probability that the car is preferably high charged when the client returns.

A developed tool assists the computations in a button click fashion. We assembled and completed three existing tools to build and analyze HPnG models. A graphical editor is used to visualize the models. Furthermore the resulting software is equipped with analyzation and simulation code to compute the measures of interest needed within this thesis.

1.2 Research approach

The aim of this thesis is to investigate the influence of grid-convenience, which is the grid operators interest, on the clients interests. We want to investigate if and in how far grid-convenience has influence on self-use and survivability of a smart home and the integrity of an EV's charging process.

New battery management strategies for smart homes that may ensure to feed in power to the grid conveniently are investigated. We compare the grid operators interests, i.e. grid-convenience with the interests of the client, i.e. self-use and survivability. The amount of power per time fed into the grid is used to assess grid-convenience. Preferably there have to be low and less gradients to relieve the grid. The self-use and survivability then are investigated by comparing their values for the original battery management strategy with the values that are computed using grid-convenient strategies.

Correct predictions of production and demand have great importance. The used grid-convenient strategies need a correct production and demand forecast to set their parameters. We use errors of the predicted production to compare the influence of the forecast quality on the original and grid-convenient battery management strategies. Note that we keep the demand fixed, since it is more controllable by the home owner.

Greater local storages are covered by involving electric vehicles batteries as an additional storage. Battery capacity is used grid-conveniently if it is offered to the grid for load balancing and frequency compensation. Within this thesis we assume that the total time, that is available to charge the car, is known. Thereby it is possible to delay the charging and thus to keep free battery capacity for a convenient use. Furthermore the grid may be able to start charging the EV in times it is little loaded. To keep a great as possible free battery capacity the charging may be finished *just-in-time*. We investigate in how far this charging strategy disadvantages the integrity, i.e. the EV is in high as possible state of charge when the client returns, which is the clients interest.

1.3 Related work

Current work on the integration of batteries and renewable power sources mainly focus on the power supply system as a whole. Grid-convenience is investigated, but usually not compared to the clients requirements. [7] discusses important topics on the integration of local battery storages into the grid. Grid-convenient battery management strategies are investigated in [8], where simulation is used to investigate in how far forecasting systems can be used to maximize the self-use. [9] also discusses the influence of grid-convenience on the self-use. They used a MATLAB based simulation model to investigate a persistence forecast management strategy and showed that this has a significantly higher potential to relieve the grid than a system that only maximizes self-use. [10] discusses how to implement battery management strategies that optimize the self-use and therewith help to reduce energy expenses. To the best of our knowledge, none of these papers take the stochastic nature of power outages and their effect on the survivability of the house into account.

The survivability of a smart home with a local storage in the presence of stochastic power outages is investigated in [3]. The trade-off in terms of cost between a flexible battery use and reserving a certain amount of battery capacity as backup is investigated in [11]. However, both papers do not consider grid-convenient battery charging strategies as presented in [12]. This thesis investigates the impact of grid-convenient battery management strategies on self-use and survivability. We incorporate different grid-convenient battery management strategies from [12] into the model presented in [3].

[13] discusses the influence of electric vehicles on the power grid. The power demand of EVs will likely have a huge impact on the stability of the power grid. Therefore grid-convenient charging strategies have to be taken into account. In [14] a real-time charging/discharging system was introduced. Therein grid-convenience shall be reached by prediction based charging and discharging. The whole system is modeled using M/G/1 queueing theory. [15] and [16] discuss the benefit of the coordinated charging of electric vehicles. They investigate stochastic programming techniques to implement coordinated charging. This thesis discusses the impact of a delayed charging, on the integrity of the charging process.

TimeNET [17] is a platform independent software that provides a graphical user interface to implement net graphs. We extended TimeNET to build and visualize HPnG models. [18] introduces the Fluid Survival Tool (FST) to model check HPnGs. It provides both basic analyzation and simulation functions. Recent analyzation algorithms are used for computations in [3]. We update the FST to the newest analyzation algorithms and extend the simulation code. TimeNET and the FST are merged to provide a software to model and analyze HPnGs.

1.4 Thesis outline

The thesis is organized as follows. Chapter 2 presents the benefit and challenges of local energy storages. We motivate the different interests of the grid operator as well as the client to use local batteries.

The measures of interest used to compare the grid operators and clients interests are presented in Chapter 3.

The HPnG formalism and the analyzation algorithms are repeated in Chapter 4. We first cover Petri Nets in general before introducing the HPnG primitives, model evolution and analyzation techniques.

Chapter 5 explains the software bundle used to compute the measures of interest. We first present the single tools and later cover how these tools are jointed into one software.

Chapter 6 presents the main battery management theory. Recent strategies to charge and discharge the battery used in smart homes are presented. To reach a more grid-convenient behavior the use of new strategies is motivated. Finally new policies are presented to be used in system modeling throughout this thesis. Since the great battery capacity offered by electric vehicles offers additional prospects to relieve the grid, furthermore a strategy to charge EVs is presented.

We investigate the influence of the new battery management strategies on smart homes in Chapter 7. We compare the grid operators needs, i.e. grid-convenience with the clients needs, i.e. self-use and survivability.

The charging of electric vehicles is investigated in Chapter 8. We assess a charging system that uses extended knowledge of the charging time to enable a coordinated use and thus to reach grid-convenience.

The thesis will be concluded in Chapter 9. Furthermore we will present proposals on possible future work.

2 Benefit and challenges regarding local energy storage

In the beginning of the energy revolution the german government subsidized the use of renewable power sources. Private clients purchased photovoltaic (PV) panels as a source of income. The price for feeding-in power to the grid was much higher than the price for purchasing it. Thus as much as possible electricity was supplied. Recently the subsidies decreased. As presented in Figure 2.1 by 2012 the *grid-parity* was reached, i.e. the price for supplying electricity was less than the price for purchasing it, for the first time. This has led to an increased interest in *self-use*. Without local energy storage, the self-use is dependent on the divergence of production and demand. The greater the time shift between production and demand, the lower is the self-use of the locally produced energy.

Recently most local batteries are used to the interest of the client, i.e. an increased self-use of the generated energy and a high survivability value of the smart home. With respect to a more balanced grid the battery may be used grid-conveniently. Grid-convenience may not be the clients interest, thus the *Kreditanstalt für Wiederaufbau* (KfW) promotes the use of grid-convenient battery management strategies for local energy storages with at most 600€/kW peak under certain restrictions [19]:

(i) at most 60% of the nominal power are allowed to be supplied to the grid (peak supply reduction)

and

(ii) the system has to be equipped with an interface which allows a remotely controlled reduction of the feed-in power.

These restrictions are according to the ones given by the German Renewable Energy Act (EEG)[20] for the implementation of new photovoltaic systems. Note that this thesis focuses on strategies that fulfill the requirement of peak supply reduction and not on remotely controlled feed-in, since the first can be implemented locally without the dependence on any external communication protocol. We investigate if and in how far grid-convenience may have an impact on self-use and survivability.

The proportion of self-use depends on the size of the local energy storage. Clients may buy greater local batteries, but great battery capacity is likewise supplied by electric vehicles. In case the local energy production does not suffice to charge an EV, the charging will be done by the grid. As presented in Figure 2.2, the number of electric vehicles (EVs) is expected to

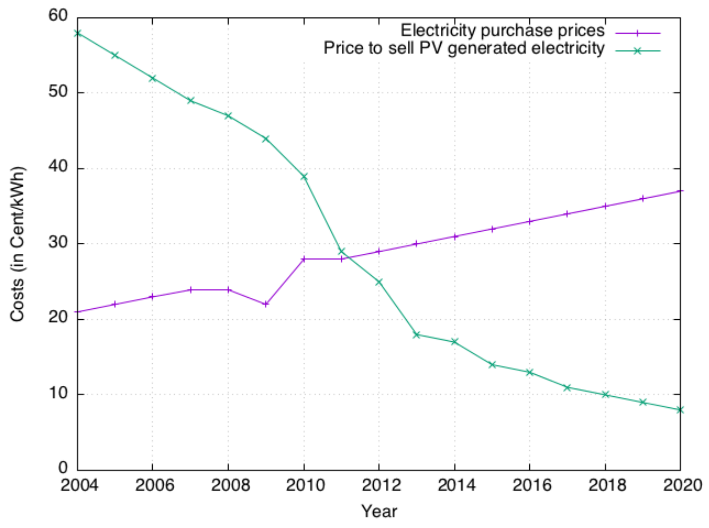


Figure 2.1. Evolution of electricity purchase and supply prices in Germany. [1]

increase enormously in the next years. Until year 2020, there should be up to 20 million EVs currently on the streets [2], whose power supply will have an huge share on the grids work. Attached to a smart home, electric cars increase the total battery capacity. Even cheaper electric cars, like the Nissan Leaf [21], offer a battery capacity of 24kWh. High end models, like the Tesla Model S [22], actually provide a battery capacity of up to 85kWh. Whenever the local production does not suffice to charge the car, the grid will be loaded. We investigate a delayed charging strategy that enables a smart grid to coordinate that charging. Again we focus on investigating in how far the grid-convenient strategies influences the clients interests.

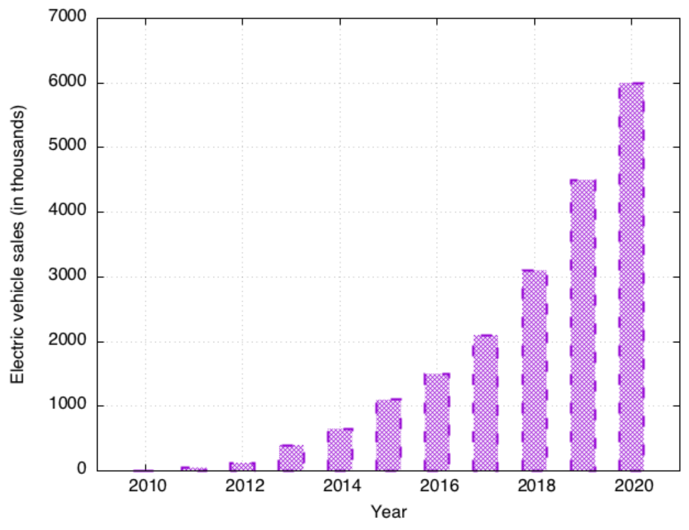


Figure 2.2. Worldwide electro vehicle sales targets. [2]

3 Measures of interest

We compare different battery charging strategies with respect to the potentially conflicting interests of the grid operator and the user. We have used the terms of measures to assess the interests so far, but now want to present them in detail. The *grid-convenience* is the main interest of the grid operator and covered in Section 3.1. Section 3.2, 3.3 and 3.4 focus on the clients interests, i.e. *self-use*, *survivability* and *integrity*.

3.1 Grid-convenience

The following definitions are inspired by [12]. Today, before a new renewable power source is connected to the grid, its *grid-compatibility* must be shown: A renewable power source is **grid-compatible** whenever it fulfills the requirements of quality, reliability and security determined by the grid-operator. Thus a grid-compatible power source does not violate the grid as soon as it is in use. A grid-compatible device does not destabalize the power grid, but not necessarily supports the stability of the power grid.

The meaning of the term grid-convenience enlarges the term grid-compatibility. **Grid-convenient** systems contribute to the grid stability during its operation. It means that a convenient system supports the grid once it gets into trouble. For instance this can be done by saving energy to be able to support the grid in load balancing. Furthermore it may be an aim to prevent the grid from high gradients, thus prohibit to forward production peaks immediately to the grid. Whenever a local storage unit is managed in a grid-convenient way, it contributes to the overall objective of a more flexible grid.

In the following we measure grid-convenience in terms of the maximum feed-in rate and the maximum gradient, i.e. the difference between two successive feed-in rates.

3.2 Self-use

As mentioned above recently the use of local power storage aims at increasing the self-use. The falling prizes for feeding-in energy to the grid cause the client to increase the self-use. The self-use is the proportion of locally used power w.r.t. the amount of locally generated power. Let S be the self-used local generated power and G be the whole amount of local generated power. The **self-use** then is defined as the proportion of S to G:

$$\text{self-use} = \frac{S}{G}.$$

Without local storage the self-use depends on the relationship between production- and demand patterns. With local battery storage, the self-use highly depends on the overall battery capacity and on the flexibility of the chosen battery management strategey. If for example a certain amount of battery capacity is reserverd as back-up capacity, less battery storage is available for self-use.

3.3 Survivability

A complete breakdown of the power supply may have a severe impact on a smart house. Power outage times are monitored by the Council of European Energy Regulators (CEER). As mentioned in [23] and [24] the system average interruption duration index (SAIDI) differs largely per country. The countries with least interruption duration, Germany and Netherlands, have an index of $16,2$ minutes respectively $33,7$ minutes. The grid fails many thousand times a year, but the SAIDI does only contain outages that persist for more than 3 minutes and thus are obviously detected by the client.

Local storage can also be used to increase the resilience of a smart house in the presence of power outages. One way of doing this is to reserve a certain amount of battery capacity to be used only when the power grid fails, as investigated in [11].

As a measure for resilience we use the so-called **survivability**, which is specified as the probability that power is continuously available during a power outage until the grid is repaired within t time units after the grid outage occurred. In the following we use *Given the Occurrence Of Disaster* (GOOD) models [25], where a failure (the power outage) is assumed to occur at a certain time a. This can be specified formally, using the so-called until operator of Stochastic Time Logic (STL) [6]:

$$\texttt{survivability} = \texttt{power\_available}\, \mathcal{U}^{[\texttt{a},\texttt{a+t}]}\texttt{grid\_on}.$$

This formula holds, when power is available continuously from time a onward until the grid is repaired before time $a + t$. Note that time a corresponds to the time of failure and is parametrized to consider the impact of different failure times during the day. The repair time of the grid is distributed according to a folded normal distribution with parameters $\mu = 0.5$ and $\sigma = 1$. The time bound t is chosen to be very large (equals the maximum time considered in the analysis), since the house can be powered as long as the property `battery_up` holds and the repair of the grid is not the main concern of the home owner.

3.4 Integrity

Whenever an electric vehicle is attached to a smart house, the battery capacity highly increases. If the EV was continuously available in times of high production, the self-use may be around 100%. Connected to a charging station, the battery should be as high as possible charged when the client returns. If the local power production does not suffice to charge the

EV, the charging will stress the grid. We assess a battery management strategy that works grid-conveniently by delaying the charging process.

In order to be able to assess the introduced charging system of electric vehicles we cover the satisfaction of the charging process. We call this **integrity** throughout this thesis. The integrity is defined as the probability that the battery is *completely* charged whenever the client returns. Note that completely does not have to mean that the battery is full. It means that, with respect to the available time to charge, the battery has a high as possible state of charge.

Again we use the a stochastic time logic until formula to model check the integrity:

$$\texttt{integrity} = \texttt{loading}\,\mathcal{U}^{[\texttt{0},\texttt{t}]}\texttt{full}.$$

The above formula holds whenever the car is completely charged before the client returns. We consider a maximal charging time of t hours. The recurrence of the client to the car will be modeled by a random distribution. The client determines his expected recurrence time and the system then creates a *just-in-time* strategy to charge the EV. The used distribution and parameters will be presented in Section 8.3.

4 Fundamentals

We use Hybrid Petri Nets with a single general one-shot transition (HPnGs), a subclass of Petri Nets (PNs), to model a smart home in this thesis. In order to understand HPnGs we first want to explore the more simple PN formalism in Section 4.1. We concentrate on a more informal presentation to introduce solely the main concepts and applications of PNs. Afterwards we introduce some PN extensions that help to converge to the HPnG formalism. A more formal description of HPnGs is then given in section 4.2.

4.1 Petri Nets

Petri Nets are a graphical paradigm for the formal description of the logical interactions among parts or of the flow of activities in complex systems. PNs have been successfully used to model systems with concurrency, synchronization or parallelism. As we will see, even these rudimentary models will play an important role when modeling smart homes. The following introduction to PNs follows [26].

4.1.1 Places, transitions and arcs

A PN is a bipartite directed graph. Two types of nodes, i.e. **places** and **transitions**, alternate on a path made up of consecutive **arcs**. The number of places and transitions is *finite* and *not zero*. An arc is directed and connects either a place to a transition or a transition to a place. Figure 4.1 shows a simple PN model with two places, two transitions and four arcs. Throughout the thesis we will denote the set of places $\mathcal{P}$ and the set of transitions $\mathcal{T}$. For example in Figure 4.1 we have $\mathcal{P} = \{\text{Grid up}, \text{Grid down}\}$ and $\mathcal{T} = \{\text{Grid failed}, \text{Grid repaired}\}$. Places usually describe the state of the system under research whereas transitions represent occurring events that may change the systems state. Place *Grid up* is said to be an **input** to transition *Grid failed. Grid down* is then the **output** to transition *Grid failed.*

4.1.2 Marking

Each place in a PN can contain an integer number of **tokens**. The vector describing the number of tokens inside each place is denoted as $\mathbf{m}$ and is called **marking**. In Figure 4.1 the *initial marking* is $\mathbf{m} = \{1, 0\}$. The marking describes the current state of the system under research. Thus if place *Grid up* contains at least one token, this means that the system is operational. The evolution of the state corresponds to an evolution of the marking, which is caused by the firing of transitions.

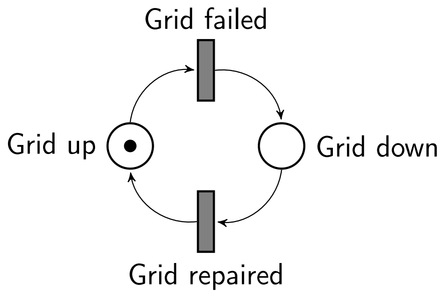


Figure 4.1. Grid status PN model.

4.1.3 Firing of a transition

As we have seen above, each transition corresponds to a certain number of input places. A transition can only fire if each of its input places contains at least one token. The transition is then said to be **enabled**. **Firing** a transitions leads to a change of the system's state. This means that from each input place a specified number of token is removed and added to each output place of the firing transition. In Figure 4.1 the transition *Grid failed* is enabled. Whenever transition *Grid failed* fires, i.e. when the system fails, the token is removed from place *Grid up* and added to place *Grid down*. This is according to the change of the system's state from *up* or *working* to *down* or *failed*.

4.1.4 Petri Net extensions

As we have seen, the original PN formalism did not convey any notion of time. But in order to investigate the performance and reliability of systems a notion of time is needed. Whenever a transition is enabled, it may not fire immediately. In the above example we want the system to be much longer in status *Grid up* than in *Grid down*. This means that transitions should have at least different discrete firing times. As we have mentioned in Section 2 to model repair procedures even general distributed firing times would be nice. Therefore two different specialisations of PNs have been introduced. **Deterministic Petri Nets** (DPNs) [27] [26] handle deterministic time events or events within a certain interval. **Stochastic Petri Nets** (SPNs) [28] have been developed, where probability distributions have been taken into account.

As we want to model smart homes, batteries definitely play an important role. Their state mainly consists of their actual charge which is represented by a continuous variable. This leads to **Hybrid Petri Nets** (HPNs) [26] that both support discrete and continuous variables. The actual level of charge can be modeled by a continuous place. Firing takes place like a continuous flow, which can be used to describe the charging and discharging of the battery.

In this section we explained PNs to introduce their main functionality and applications. We have seen that the simple PN formalism is not sufficient to model smart homes. We need a notion of time as introduced by DPNs and SPNs and continuous variables as introduced with HPNs. Below HPnGs will be introduced, in a more formal way than PNs above, that merge DPNs, SPNs and HPNs characteristics to be able to model and analyze smart homes.

4.2 Hybrid Petri Nets with general one-shot transitions

As introduced in [5] and extended in [3] HPnGs are a Hybrid Petri Net formalism having generally distributed transitions that move discrete tokens. HPnGs are used to model highly complex systems containing both discrete and continuous quantities. Moreover, random failures can be modeled by HPnGs. The discrete part is used to control the continuous one. In case of smart homes we need the discrete part to model the status of the grid, the production or demand pattern and the battery management strategy. The continuous part is used to model the battery. First the model primitives are introduced in Section 4.2.1 before explaining the detailed model definition in Section 4.2.2. Even though we will not completely introduce the whole evolution procedure we will give an insight in Section 4.2.3. The main objective of this Section is to understand the HPnG formalism to thus be able to model smart homes.

4.2.1 Model primitives

The HPnGs formalism is build up upon a standard DPN formalism. The set of **places** $\mathcal{P}$ contains two disjoint sets of **discrete** ($\mathcal{P}^D$) and **continuous** ($\mathcal{P}^C$) places describing the state of the system. As mentioned in Section 4.1 a discrete place may contain an integer number of tokens, while a continuous place is assigned with a real value, representing the amount of fluid in it.

Transitions will trigger a change in the state of the system. The set of transitions $\mathcal{T}$ consists of five different types fo transitions. **Deterministic** ($\mathcal{T}^D$) transitions fire after a constant time from the moment they became enabled. As the name says **immediate** ($\mathcal{T}^I$) transitions fire just in the moment they became enabled. They can be considered as deterministic transitions with firing time equal to 0. **General** ($\mathcal{T}^G$) transitions fire according to a random delay. Within the HPnG formalism they can only fire once. All these three transition types are changing the discrete part of the system. **Continuous** transitions ($\mathcal{T}^C$) change the continuous part of the system. Moreover, a continuous transition can be **static** or **dynamic**, meaning that it will either fire with constant rate or dynamically depending on the rates of other static continuous transitions.

By the set of **arcs**, $\mathcal{A}$, it is characterized how transitions and places are related to each other. **Discrete** arcs ($\mathcal{A}^D$) work as described in Section 4.1. They remove a certain number of tokens from discrete input places and add a certain number to discrete output places whenever a corresponding transition fires. **Continuous** arcs ($\mathcal{A}^C$) connect continuous places and continuous transitions. Whenever such a transitions fires, a defined amount of fluid is moving throughout the arc from its input to the output place. **Guard** arcs ($\mathcal{A}^G$) are a special type of arc. They do not change any marking, they only enable the dedicated transitions. Guard arcs connect discrete places to all kinds of transitions and also continuous places to all but continuous transitions. Figure 4.2 shows an overview of the graphical representation of the above introduced HPnGs primitives.

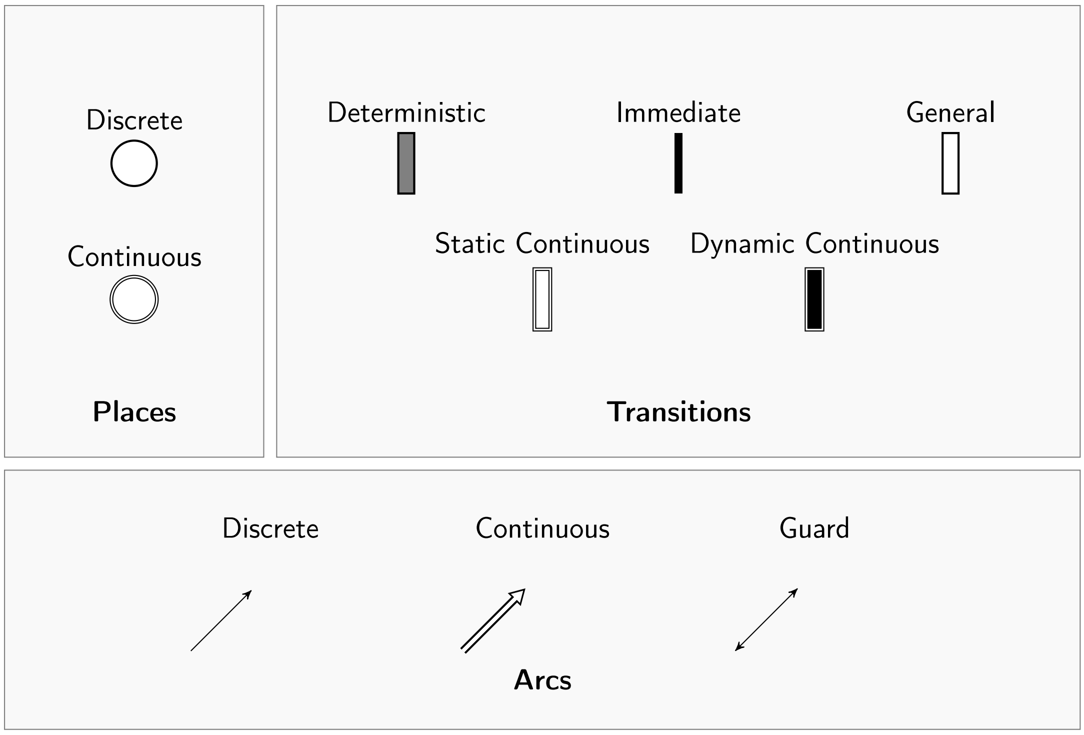


Figure 4.2. Graphical representation of HPnGs primitives.

4.2.2 Model definition

A HPnG is formally defined as a tuple $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathbf{m}_0, \mathbf{x}_0, \Phi)$. We have already introduced the set of places, transitions and arcs in Section 4.2.1. As seen above $\mathbf{m} = \{m_1, ..., m_{|\mathcal{P}^D|}\}$ denotes the discrete marking, a vector containing an integer value $m_i \in \mathbb{N}^{\geq 0}$ for each of the discrete places. The continuous marking $\mathbf{x} = \{x_1, ..., x_{|\mathcal{P}^C|}\}$ thus contains an real value $m_i \in \mathbb{R}^{\geq 0}$ for each of the continuous places. The initial marking of a HPnG is composed of the discrete initial marking $\mathbf{m}_0$ and continuous initial marking $\mathbf{x}_0$.

The tuple Φ contains 9 functions $\Phi = \{\phi_b^{\mathcal{P}}, \phi_w^{\mathcal{T}}, \phi_p^{\mathcal{T}}, \phi_d^{\mathcal{T}}, \phi_f^{\mathcal{T}}, \phi_g^{\mathcal{T}}, \phi_w^{\mathcal{A}}, \phi_s^{\mathcal{A}}, \phi_p^{\mathcal{A}}\}$ that assign parameters to places, transitions and arcs. An upper boundary to continuous places is assigned by $\phi_b^{\mathcal{P}} : \mathcal{P}^C \to \mathbb{R}^+ \cup \infty$. Every continuous place has an implicit lower boundary of 0.

The weight is defined by $\phi_w^{\mathcal{T}} : \mathcal{T}^I \cup \mathcal{T}^D \to \mathbb{R}^+$ and the priority by $\phi_p^{\mathcal{T}} : \mathcal{T}^I \cup \mathcal{T}^D \to \mathbb{N}$ for immediate and deterministic transitions. These are used to resolve conflicts between transitions that are supposed to fire at the exact same time. The constant firing time for deterministic transitions is defined by $\phi_d^{\mathcal{T}} : \mathcal{T}^D \to \mathbb{R}^+$, similarly a constant nominal flow rate is assigned to every continuous transition by $\phi_f^{\mathcal{T}} : \mathcal{T}^F \to \mathbb{R}^+$. General transitions are assigned with a cumulative density function (CDF) $\phi_g^{\mathcal{T}} : \mathcal{T}^G \times \mathbb{R}^+ \to [0, 1]$ where $\phi_g^{\mathcal{T}}(T_k^G, \tau)$ denotes the probability that T_k^G fires before time τ.

The number of tokens required or produced by discrete and guard arcs is specified by the weight $\phi_w^{\mathcal{A}} : \mathcal{A} \to \mathbb{R}^+$ that is assigned to all types of arcs. For continuous arcs the weight is multiplied with the transitions rate. This can be used to specify whether different portions of an transition are used from different components but at the same rate. Like deterministic and immediate transitions fluid arcs also have a priority $\phi_p^{\mathcal{A}} : \mathcal{A}^f \to \mathbb{N}$ and a share $\phi_s^{\mathcal{A}} : \mathcal{A}^f \to \mathbb{R}^+$ to resolve conflicts in the fluid flow.

4.2.3 Model evolution

In order to understand how an HPnG model evolves we want to introduce simple evolution basics in the following. We would rather give an overview as an detailed explanation of the used algorithms.

Concession The status of the HPnG evolves according to the firing rules of its transitions. As we have mentioned above transitions correspond to input places. Thus in case of a discrete transition $T_i^D \in \mathcal{T}^D$ connected with an arc $A_i \in \mathcal{A}^D \cup \mathcal{A}^G$, T_i^D has concession whenever all its input places contain at least $w_i = \phi_w^{\mathcal{A}}(A_i)$ tokens. Thus w_i is the weight corresponding to the given arc. A continuous transitions $T_i^F \in \mathcal{T}^D$ can be connected with a guard arc $A_i \in \mathcal{A}^G$ to a discrete place. Again T_i^F then has concession whenever the discrete place contains at least $w_i = \phi_w^{\mathcal{A}}(A_i)$ tokens. A continuous transitionx $T_i^F \in \mathcal{T}^F$ connected to a fluid place has concession whenever the current amount of fluid is greater than the weight of the associated arc. In case a transitions has concession it may be enabled and thus fires.

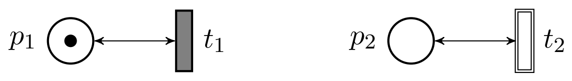


Figure 4.3. HPnG model considering the concession of transitions.

Figure 4.3 considers the most commonly used examples to control transitions in HPnG models. Let both arcs have weight $w = 1$. Thus since place *p1* contains one token, transition t_1 has concession. t_1 will always have concession in case p_1 contains more than one token. Since p_2 contains no token at all, t_2 does not have concession.

Discrete places and discrete transitions are used to model the evolvement of the discrete part of the system. As mentioned above, the discrete part is used to control the continuous one. Thus to state that a continuous transition, for instance a pump, works is done as shown with place p_2 and transition t_2.

Enabling Whenever the continuous transition $T_j^F \in \mathcal{T}^F$ has concession it continuously transport fluid with rate $\phi_f^{\mathcal{T}}(T_j^F)$ along its fluid arcs. Thus fluid transitions are enabled if they have concession. To take care that a fluid place can not trespass its boundaries a *rate adaption algorithm* is used. Whenever a fluid place reaches its lower boundary the outflow has to match the inflow. Likewise the inflow has to match the outflow whenever the upper boundary is reached.

Discrete transitions that have concession may be enabled. Whenever a immediate transition has concession, the marking is called **vanishing**. This means that immediate transitions are enabled and have precedence above deterministic and general transitions. In a vanishing marking, deterministic and general transitions are disabled. If no immediate transitions are enabled, the marking is called **tangible**. In a tangible marking *race policy* is used to determine which transition will be enabled next. Therein simply the clock values were considered to determine which transition can fire next. A clock preserves the total time a timed transition is enabled so far.

In Figure 4.4 a) both t_1 and t_2 have concession. Since t_1 is a immediate transition, it is enabled and will fire next. Thus this marking is vanishing. The deterministic transition t_2 is always disabled and will never fire.

Conflict resolution In case several transitions are scheduled to fire at the exact same time, priorities and weights are used to resolve this conflict. Let $\mathcal{C}$ be a set of all transitions that have concession and do have the exact same priorities. The relative firing probability $p(T_i)$ of each of these transitions $T_i \in \mathcal{C}$ then is calculated as:

$$p(T_i) = \frac{\phi_w^{\mathcal{T}}}{\sum_{\forall T_i \in \mathcal{C}} \phi_w^{\mathcal{T}}(T_i)}.$$

This is exactly the same for immediate and deterministic transitions. However for deterministic transitions we have to consider the so called *residual firing time* τ^T to calculate the set $\mathcal{C}$. For a deterministic transition $T_i^D \in \mathcal{T}^D$, the residual firing time is the difference of its firing time $\phi_d^{\mathcal{T}^D}$ and the evolved clock time c_i.

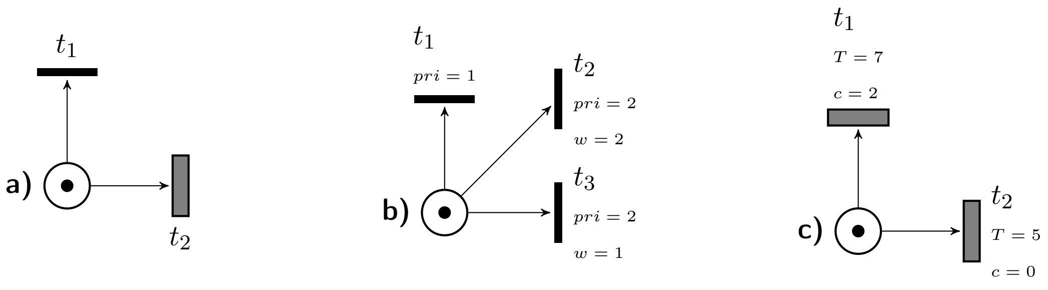


Figure 4.4. HPnG model for enabling and conflict resolution.

In Figure 4.4 b) t_1, t_2 and t_3 have concession but due to the priorities only t_2 and t_3 are enabled. As presented above the probability that t_2 fires first is 1/3 and that t_3 fires first is 2/3. c) shows that whenever two deterministic transitions are enabled, the residual time to fire (rtf) has to be observed. Transitions t_1 and t_2 have a rtf of 5 and thus fire at the exact same time. Again the weights are used to calculate the relative firing probability of each of the two enabled transitions.

Firing rules Whenever a discrete transition fires, only the discrete marking is changed. Let w_i be the weight of the arc that connects the input place $P_i \in \mathcal{P}$ with an enabled transition and w_o be the weight of the arc connected with the output place $P_o \in \mathcal{P}$. After firing, w_i tokens are be removed from P_i and w_o tokens are added to P_o. In case a deterministic transition fires, its clock value is set to 0. The clock value for general transitions is set to -1 after firing. If an immediate transition fires only the marking $\mathbf{m}$ will change and the clocks will not evolve.

State of the model The state of a HPnG is identified by the tuple $\Gamma = (\mathbf{m}, \mathbf{x}, \mathbf{c}, \mathbf{g})$, where $\mathbf{m}$ and $\mathbf{x}$ denote the initial and continuous marking as introduced above. We have seen that each deterministic transition fires after a specified time. The vector $\mathbf{c} = (c_0, ..., c_{|\mathcal{T}^D|})$ stores the time c_i for every deterministic transition $T_i^D \in \mathcal{T}^D$ that has passed since it has been enabled. The clock value will be preserved whenever a transitions is disabled. The clocks are only resetted when the transition fires. Likewise the enabling time g_i of each general transition $T_i^G \in \mathcal{T}^G$ is stored in the vector $\mathbf{g} = (g_0, ..., g_{|\mathcal{T}^G|})$. Since general transitions can only fire once, the value -1 is used to indicate that the corresponding transition has already fired.

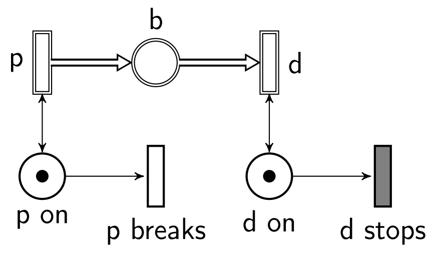


Figure 4.5. HPnG model considering discrete and continuous parts.

In Figure 4.5 both continuous transitions p and d are enabled in the initial marking, since *p on* and *d on* each contain one token. Let p be a production and d be a demand that both have rate $r = 4$. Thus as long as p and d are enabled, the amount of fluid in the continuous place does not change. The production breaks according to a general transition *p breaks* and the demand stops at a fixed point in time according to the deterministic transition *d stops*. Let the demand stop at $t = 5$. Moreover we assume that the general transition fires at $t = 2$. The initial amount of fluid in the continuos place will be 15. Hence at time $t = 0$ the initial marking will be:

$$\Gamma = (\mathbf{m} = \{1, 1\}, \mathbf{x} = \{15\}, \mathbf{c} = \{0\}, \mathbf{g} = \{0\}).$$

Until time $t = 2$ only the clocks $\mathbf{c}$ and $\mathbf{g}$ will evolve. At time $t = 3$ the transition *p breaks* has fired:

$$\Gamma = (\mathbf{m} = \{0, 1\}, \mathbf{x} = \{11\}, \mathbf{c} = \{3\}, \mathbf{g} = \{-1\}).$$

Now the amount of fluid decreases with drift $d = 4$ until the demand will also stop at time $t = 5$. Thus at time $t = 6$ the final marking will be:

$$\Gamma = (\mathbf{m} = \{0, 0\}, \mathbf{x} = \{3\}, \mathbf{c} = \{0\}, \mathbf{g} = \{-1\}).$$

From now on nothing will change since no transition has concession.

According to the HPnG model evolution presented above, a *simulation* of a model can be done by observing the state of the model for an increasing time. For example a HPnG can be implemented and simulated for a certain time t. One possibility to simulate a trace of the model is to identify all possible events. An event is triggered whenever a transition fires or when a continuous place reaches its boundary. Thus to simulate a trace of the model, the next event has to be calculated. Between two consecutive events the discrete marking remains unchanged and the continuos marking changes linearly. We may use simulation to investigate the state of charge of the battery in a smart home.

If we want to investigate if certain properties hold on the model at hand, for example: *Does a fluid place get drained?*, we use *Model Checking* which is presented in Section 4.4. Model Checking algorithms use a *state representation* of all reachable states to perform the analyzation. We introduce two commonly used methods to efficiently generate all reachable states in the following Section.

4.3 HPnG State Representation

As mentioned above, firing of a transition or reaching a fluid place boundary are events that separate different system states. We will shortly present two different ideas to calculate all reachable states. In both algorithms a conditioning and deconditioning technique is applied.

Conditioning on the firing time of the general transition means that the evolution of the system is distinguished by the fact if the general transition already fired. This results in two different suitable state representations as presented in Section 4.3.1 and Section 4.3.2. *Deconditioning* is the integration over the probability density function for intervals derived from the state representation.

4.3.1 Parametric reachability analysis

The original approach to compute all reachable states of a HPnG model is called *parametric reachability analysis* and was introduced in [5]. Each state is represented as a so called parametric location. A parametric location is an extension of a HPnG state as presented in Section 4.2.3. All parametric locations are represented with a treelike structure. Every node in the tree is a parametric location and every edge represents an event in the system evolution. As mentioned above the state-space is explored by the occurrence of events.

To perform the deconditioning on the computed parametric location tree, a parametric location contains a time interval that states for which possible firing time of the general transition the location is meaningful.

4.3.2 Region-based analysis

The region-based analysis was introduced in [29]. It is an alternative to the parametric reachability analysis that is exact and decreases the computation time. The whole state space is separated by events into regions that have an equivalent discrete marking. A 2-dimensional representation is used to visualize this separation.

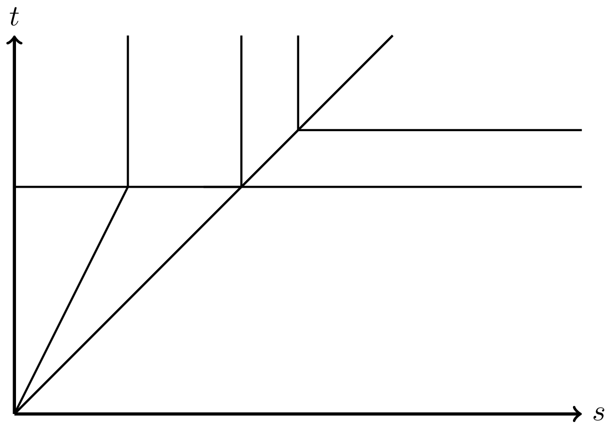


Figure 4.6. Graphical representation of a STD.

Again a conditioning on the firing time of the general transition is used. Let s be the time the general transition is fired. The state of the system can be determined for all future times t. Hence, in order to characterize the systems state, a 2-dimensional diagram called Stochastic Time Diagram (STD) is used having s on the x-axis and t on the y-axis. The line $t = s$ separates the stochastic area from the deterministic area. Above the line $t = s$ is the stochastic area which means that $t > s$ and thus the general transition has already fired. The area below $t = s$ is the deterministic area. Since $t < s$ the general transition is not fired there.

The deterministic area is further partitioned by events represented as a horizontal line in the STD. Events take place at a specified point in time and are independent of s in the deterministic area. As stated by Proposition 1 in [29] even in the stochastic area the partitioning is done with linear lines and only changed by an event. Figure 4.6 shows an example of the graphical representation of a STD. Each point in the STD is denoted by $\Gamma(s, t)$ and represents a unique state of the system. Since events separate different system states, each regions represents states with identical discrete marking.

Within the conditioning phase a random variable is fixed to retrieve sets of deterministic evolution. As presented above, the STD is the result from the conditioning algorithm. Deconditioning assigns probabilities to sets of evolution according to a probability density function (pdf). For a arbitrary time $t = \tau$ all intersecting regions are computed. These intersections represent the intervals used to integrate over the pdf. This procedure is further explained in Section 4.4.2. Generally conditioning and deconditioning leads to transient probabilities for a HPnG model, i.e. the probabilities to be in a certain state at a specified point in time.

4.4 Model Checking HPnGs

A *model checker* checks automatically if a model of a system meets a given requirement. As mentioned above we now want to investigate if a HPnG model fulfills certain properties. Figure

4.7 illustrates the general model checking procedure. We have to formalize the requirements into a property specification that can be checked on the model before the model checking can proceed.

A logic called Stochastic Time Logic (STL) is used to formalize the requirements for HPnG models. A typical model checker returns a verdict, i.e. a yes/no answer. Moreover, if the verdict is no, a counter-example can be generated. The model checker presented in this Section is based on the STD created by region-based analysis. As we have seen, deconditioning results in a probability and not in a verdict. Thus a probability operator is wrapped around the STL formula to compare the resulting probability with a given probability bound to compute the verdict. Furthermore we are able to use the computed probabilities as a result of the model checking.

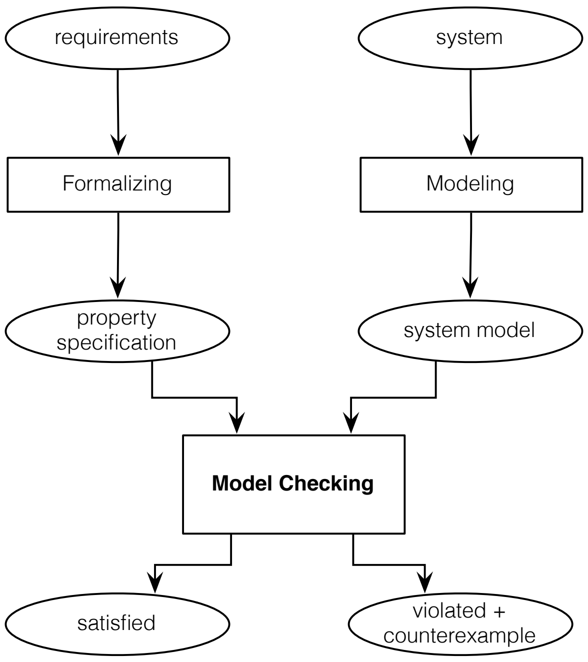


Figure 4.7. Model Checking in general.

4.4.1 Stochastic Time Logic

The Stochastic Time Logic (STL) was introduced in [6] to allow the investigation of both state-based and time-bounded path-based properties on a HPnG model. Therefore it enhances the logic presented in [29] with an until operator. In the following definition the syntax of the STL will be presented:

Definition 4.1 (The Syntax of STL)**.** An STL formula Ψ is defined as:

$$\Psi = tt \quad | \quad x_{\mathcal{P}} \leq c \quad | \quad m_{\mathcal{P}} = a \quad | \quad \neg\Psi \quad | \quad \Psi \wedge \Psi \quad | \quad \Psi \mathcal{U}^{[0,T_1]} \Psi,$$

where $T_1, c \in \mathbb{R}^{\geq 0}; a \in \mathbb{N}^{\geq 0}$.

The state-based properties in STL are atomic properties. Atomic properties are indivisible and thus not constructed by other properties. As stated above, the time to check t needs to be specified when investigating state-based properties. The first state-based property ($x_{\mathcal{P}} \leq c$) is a continuous atomic property. $x_{\mathcal{P}}$ is the marking of the continuous place $\mathcal{P}$. The property is satisfied whenever the marking of $\mathcal{P}$ at time t is smaller or equal to c. The discrete atomic property ($m_{\mathcal{P}} = a$) if the discrete place $\mathcal{P}$ contains at least a token at time t. Properties can be negated ($\neg$) and constructed by a conjunction ($\wedge$) of two STL properties.

A path-based property in STL is constructed using an until operator $\Psi \mathcal{U} \Phi$ with a time-bound from 0 to T_1 and two STL properties Ψ and Φ. The until operator checks the probability that only the property Ψ holds until the property Φ holds in the specified time bound.

Again consider the HPnG model from Figure 4.5. Let the continuous place b model a battery and p be the production from a renewable power source. Let d be the demand of the smart house at hand. We are now able to formalize model checking questions on that model as follows:

Question	**STL formula**
Pump broken?	p on $= 0$
Battery empty?	$b \leq 0$
Charge of battery greater than 300 until the production breaks for the time interval $[0, 5]$?	$\neg b \leq 300 \quad \mathcal{U}^{[0,5]} \quad$ p on $= 1$

Table 4.1. Example of STL formulas for model checking HPnGs.

All the requirements stated in Table 4.1 are associated with a defined time to check t. Thus the probabilities are computed specifically for time t. The conditioning and deconditioning technique presented in Section 4.3 computes a Stochastic Time Diagram. With a STL formula certain properties are defined that can be checked for an arbitrary time in the STD. This leads to probabilities according to the probability density function associated with the firing time of the general transition. We will use STL formula for survivability analyzation of a smart home in this thesis.

Actually we know *what* requirements can be checked on a HPnG model, but we do not know *how* the model checking algorithms work on the STD so far. In order to understand the model checking procedure, we shortly want to introduce the algorithms in the next section.

4.4.2 Model Checking algorithm

The following presented model checking algorithm was first introduced in [6]. It takes a HPnG model and a STL specification as input and results in a probability that the formula holds. In this Section the basic concepts of model checking state-based and path-based STL formula is presented. More details are provided in [29] and [6].

Model Checking state-based STL formula

The general procedure is based on the conditioning and deconditioning technique presented in Section 4.3.2. First a STD is generated from the HPnG model at hand. As mentioned above, every model checking procedure works on a certain time to check τ. As previously presented each region represents states $\Gamma(s,t)$ with an identical discrete marking. According to Definition 4.2 the algorithm first has to find all regions that satsify the given STL property.

Definition 4.2 (Statisfaction of STL formula)**.**

$$\begin{array}{ll}
\Gamma(s,t) \models^{s,t} tt & \forall t, s, \\
\Gamma(s,t) \models^{s,t} x_{\mathcal{P}} \leq c & \textit{iff}\ \Gamma(s,t).x_{\mathcal{P}} \leq c, \\
\Gamma(s,t) \models^{s,t} m_{\mathcal{P}} = a & \textit{iff}\ \Gamma(s,t).m_{\mathcal{P}} = a, \\
\Gamma(s,t) \models^{s,t} \neg\Psi & \textit{iff}\ \Gamma(s,t) \not\models^{s,t} \Psi, \\
\Gamma(s,t) \models^{s,t} \Psi_1 \wedge \Psi_2 & \textit{iff}\ \Gamma(s,t) \models^{s,t} \Psi_1 \wedge \Gamma(s,t) \models^{s,t} \Psi_2, \\
\Gamma(s,t) \models^{s,t} \Psi_1 \mathcal{U}^{[0,T_1]} \Psi_2 & \textit{iff}\ \exists \tau \in [t, t+T_1] : \Gamma(s,t) \models^{s,t} \Psi_2 \wedge (\forall \tau' \in [t,\tau] : \Gamma(\tau', s) \models^{s,t} \Psi_1).
\end{array}$$

The algorithm has to determine the satisfaction of the STL property at a certain time to check t. As stated in Defintion 4.3 the set of satisfaction intervals has to be computed to get the intervals for the deconditioning technique.

Definition 4.3 (Set of satisfaction Intervals)**.** An interval I satisfies property Ψ at time t whenever the following condition holds:

$$I_{\Psi} \models^{t} \Psi \ \textit{iff}\ \forall s \in I : \Gamma(s,t) \models^{s,t} \Psi.$$

The set of satisfaction intervals $Sat^t(\Psi)$ is defined as the set of all intervals satisfying Ψ at time t:

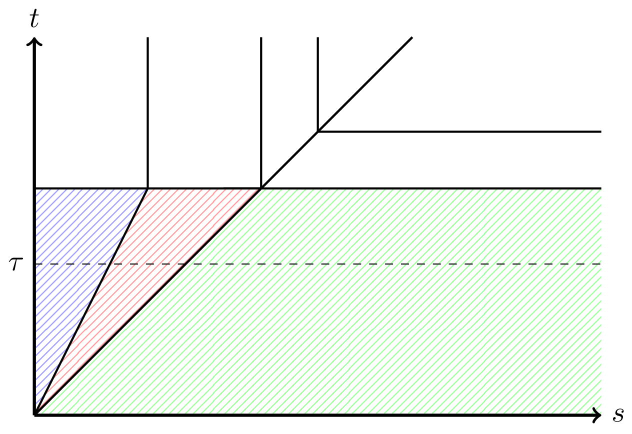
$$Sat^t(\Psi) = \{I_{\Psi} : I_{\Psi} \models^{t} \Psi\}.$$

According to Definition 4.4 the probability can be computed by deconditioning with the probability density function $g(s)$ specified along with the general transition.

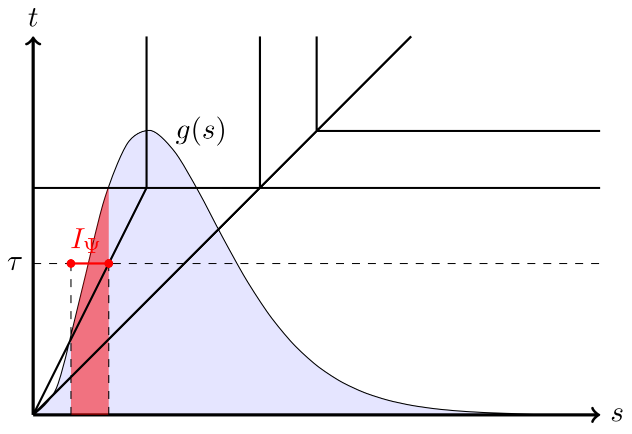
Definition 4.4 (Probability computation)**.** The probability that condition Ψ is satisfied at time t can be computed as:

$$Prob(\Psi, t) = \sum_{I_{\Psi} \in Sat^t(\Psi)} \int_{I_{\Psi}} g(s)\,ds.$$

Since model checking generally results in a verdict, in [6] a probability operator was introduced that simply tests whether the computed probability matches a certain probability



a) Find intersecting regions.



b) Compute satisfaction interval for deconditioning.

Figure 4.8. Model Checking example - Working on a STD

threshold. We will omit the definition of the probability operator because we are interested in probability values.

Figure 4.8 illustrates the main steps taken to model check a HPnG model. First all regions are determined intersecting the line $t = \tau$ as shown in **a)**. Given a STL formula Ψ, the algorithm determines in which region Ψ holds. Discrete atomic properties either hold in a entire region or not at all, however continuous atomic properties may hold in a certain part of a region. Let the blue lined region fulfill a continuos property in Figure 4.8 **b)** only for the interval I_Ψ, thus $Sat^t(\Psi) = \{I_\Psi\}$ Finally the probability is calculated by integrating over the probability density function $g(s)$ for the set of Intervals $Sat^t(\Psi)$.

Model Checking path-based STL formula

The main idea for checking until formula $\Phi_1 \mathcal{U}^[0, T_1]\Phi_2$ is to find all intervals that satisfy Φ_1 at the starting time. The initial intervals are refined by omitting those parts which violate Φ_1. This will be continued by either reaching the time bound T_1 or a time point in $[0, T_1]$ which satisfies Φ_2. The algorithm for model checking the until formula is working on a set of intervals and a set of sets. Hence to introduce the satisfaction set of the until formula, first the intersection operator and the complement operator between set of sets are presented in Definition 4.5. Afterward the set of satisfaction intervals for bounded until formula is introduced in Definition 4.6

Definition 4.5 (Set of sets)**.** Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two set of sets, the intersection operator and the complement are defined as follows:

$$\begin{aligned} &S \in \mathcal{S}_1 \sqcap \mathcal{S}_2 && \textit{iff } \exists S_1 \in \mathcal{S}_1, S_2 \in \mathcal{S}_2 : S = S_1 \cap S_2, \\ &S \in \neg\mathcal{S}_1 && \textit{iff } \forall S' \in \mathcal{S}_1 : S' \cap S_1 = \emptyset \wedge \nexists S'' \in \neg\mathcal{S}_1 : S'' \subset S. \end{aligned}$$

According to the above definition, the union operator can be derived as $\mathcal{S}_1 \sqcup \mathcal{S}_2 = \neg(\neg\mathcal{S}_1 \sqcap \neg\mathcal{S}_2)$. Intuitively, the union operator merges two sets ensuring that the result is maximal and contains no duplicates.

In order to model check the bounded until operator for each possible firing time of the general transition at starting time point t_0, a time point τ between t_0 and $t_0 + T_1$ has to be found, where Φ_2 holds and before which Φ_1 is not violated:

Definition 4.6 (Set of satisfaction intervals for until formula)**.**

$$Sat^t(\Psi_1 \mathcal{U}^{[T_1,T_2]} \Psi_2) = \sqcup\{I \subseteq \mathbb{R}_{\geq 0} | \exists \tau \in [t_0 + T_1, t_0 + T_2] : I \models^{\tau} \Phi_2 \wedge \forall t' \in [t_0, \tau] I \models^{t'} \Phi_1\}.$$

Considering the linear boundaries of a region, the validity area of a discrete atomic and continuous atomic property is a polygon. Thus the algorithm to model check a time bounded until formula works on polygons to determine the deconditioning intervals. The deconditioning procedure still remains the same. More details on Model Checking until formula can be found in [29].

We presented the HPnG formalism in this chapter. We have seen how the primitives are graphically represented and explained the meaning of places, transitions and arcs. We introduced the model evolvement to understand how HPnG models work. Furthermore we have seen how to observe different system states by model simulation. Finally we have shown the main ideas of model checking HPnGs and thus introduced all the fundamentals needed to use HPnGs for the investigation of smart home.

5 Tool development

The modeling of real-world systems is a complex task which often results in very large models. A software tool is needed to model and analyze HPnGs. In the following we present a developed software that provides a GUI to implement HPnG models and allows to analyze and simulate them. Actually no GUI is provided to implement HPnGs. TimeNET [17] is a tool used to visualize the implementation of net graphs. We extend TimeNET to be a WYSIWYG[1] editor to create HPnG models.

The current software used to analyze HPnGs is a console application. The Fluid Survival Tool [30] (FST) provides a GUI based model checker and simulator for HPnGs, but not for the latest analyzation code. Therefore we merge the console applications code with the FSTs code and hereby update the FST to the latest version. It is the aim to connect TimeNET with the FST to provide one single software used to create and analyze HPnGs. A modular implementation may make it easy to maintain and extend the software. The software may not only be used for this thesis but for general purposes in academic teaching such as project seminars. In conclusion the development is done in three steps:

1. Implement a TimeNET based WYSIWYG editor for HPnGs.
2. Update the Fluid Survival Tool.
3. Connect TimeNET and the FST.

5.1 GUI editor to create HPnGs

Currently HPnGs are represented by a so called *.m-file*. It is a text-based representation that specifies all transitions, places and arcs of a HPnG model. TimeNET [17] is a platform independent software for the performability evaluation with stochastic and colored petri nets. It includes a graphical user interface to implement net graphs. We extend the TimeNET editor to support the implementation of HPnG models and to replace the old *.m-files*. The extension of TimeNET is based on the implementation of an xml schema definition (xsd) representing the new formalism. The graphical representation of each HPnG element then is implemented within xml files. Lastly the implementation is enclosed by extending the java code. These steps are very straight-forward and presented in detail in [31]. In the following we present the main steps taken to implement the HPnG formalism within TimeNET.

[1]What you see is what you get.

5.1.1 Implementing the HPnG formalism

The xml schema definition of a formalism in TimeNET includes the main description of the occurring elements and their properties such as their type or multiplicity. Every element is implemented by extending a base type. For a detailed description of all used types please again refer to [31]. Generally the HPnG formalism, as presented in Chapter 4, consists of places, transitions and arcs. In the following we briefly present the implementation of a HPnG xml schema definition only containing a continuous place element.

As presented in Figure 5.1 the HPnG net class is created by extending the *NetType* base type. For the sake of simplicity the example consists of only one element which is called *continuousPlace* in this case. In general the *xsd:sequence* contains all the elements of the formalism at hand, i.e. all types of places, transitions and arcs. The extension of the *NetType* determines the multiplicity and type of an element and not its properties. The type *ContinuousPlaceType* is not a standard type and therefore needs to be specified in detail.

The principle of generalization is very helpful to prohibit code duplicates. All place elements may contain an attached label in the interface that presents their identity. A general *PlaceType* contains this label and can be used as the base type of all the place elements. The *LabelType* is not a standard type and needs to be specified again. This implementation is omitted here.

Lastly the implementation of the *ContinuousPlaceType* contains the properties of a continuous place. It is an extension of the base type *PlaceType* and thus contains a label. In addition a continuous place contains the properties *capacity* and *level* which both are standard float values. Well-known base types such as integer, double, string and even lists can be used directly as element property types.

An xml schema definition, as shown in Figure 5.1, describes elements and their properties but it does not determine how these elements are presented in the GUI. The appearance is implemented in additional xml files which will be presented in the following section.

5.1.2 Graphical representation of HPnG elements

The appearance of an element is specified in a dedicated xml-file. Its name has to be equal to the name of the element it is describing. Thus in case of the continuous place example the name has to be *continuousPlace*. A custom language based on xml is used to describe graphical elements.

As shown in Figure 5.2 the xml-file consists of elements that represent graphical forms. A *segment* is a line segment. Together with a specified width and angle of 360 degree it creates a circle that is filled with certain color. In the example in 5.2 the four circles are filled with black and white color and create an continuous place element that consists of two nested circles. Besides the type *segment* a *variableRectangle* type can be used to draw an arbitrary square.

We have added an xml-file for each custom element of the HPnG formalism. To show elements in the toolbar or within tooltips we have to provide small images. Again these images are named equally to the associated element and placed in the same folder than the xml-file.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://wwu.de/TimeNET/schema/HPnG"
  xmlns="http://wwu.de/TimeNET/schema/HPnG"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="builtin/net-objects.xsd"/>
  <xsd:element name="net" type="HPnGNet"/>

  <xsd:complexType name="HPnGNet">
    <xsd:complexContent>
      <xsd:extension base="NetType">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="0"
            name="continuousPlace" type="ContinuousPlaceType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="PlaceType">
    <xsd:complexContent>
      <xsd:extension base="NodeType">
        <xsd:sequence>
          <xsd:element maxOccurs="1" minOccurs="1" name="label"
            type="LabelType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="ContinuousPlaceType">
    <xsd:complexContent>
      <xsd:extension base="PlaceType">
        <xsd:attribute name="capacity" type="xsd:float" default="0.0"/>
        <xsd:attribute name="level" type="xsd:float" default="0.0"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

Figure 5.1. XML schema definition of an abstracted HPnG model.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graphics>
  <segment color="#000000" selectedColor="#316AC5" extensionAngle="360"
    height="40" startAngle="0" width="40" x="0" y="0"/>
  <segment color="#FFFFFF" selectedColor="#FFFFFF" extensionAngle="360"
    height="36" startAngle="0" width="36" x="0" y="0"/>
  <segment color="#000000" selectedColor="#316AC5" extensionAngle="360"
    height="32" startAngle="0" width="32" x="0" y="0"/>
  <segment color="#FFFFFF" selectedColor="#FFFFFF" extensionAngle="360"
    height="28" startAngle="0" width="28" x="0" y="0"/>
</graphics>
```

Figure 5.2. GUI description of a continuous place within TimeNET.

Additional menu entries are likewise added by implementing them in an xml-specification. We do not discuss that here but again refer to [31].

5.1.3 Customizing java classes

Each element that is implemented within the *NetType* extension within the xsd needs to have a java class representation. Furthermore a java class is needed that implements the general net class, i.e. HPnG in this case. Thus for the example in Figure 5.1. at least two classes are needed, *hpng.java* and *continuousPlace.java*. Again the class is named equally to the corresponding element.

As presented in Figure 5.3 the code is used to set advanced properties, like the elements that a continuous place can connect to. The *continuousPlace* class extends class *Place*. This shows that general inheritance hierarchies can be used to structure the code. Each of the HPnG elements thus are likewise implemented as java classes to implement their specific behaviour.

The *hpng.java* class is used for implementing the general functionality of the current editor instance. In this class additional menu items and corresponding dialogs are implemented. The *hpng.java* is the main class which is first called when a new HPnG model is created. Hence this class is used to set all needed parameters, to instantiate model values and set additional configurations.

This first development phase resulted in an extended TimeNET which enables to implement HPnG models. The complete editor is presented in the following section.

5.1.4 Extended TimeNET editor

This section shows the extended TimeNET editor. We present the GUI as well as how to implement HPnG models within the editor.

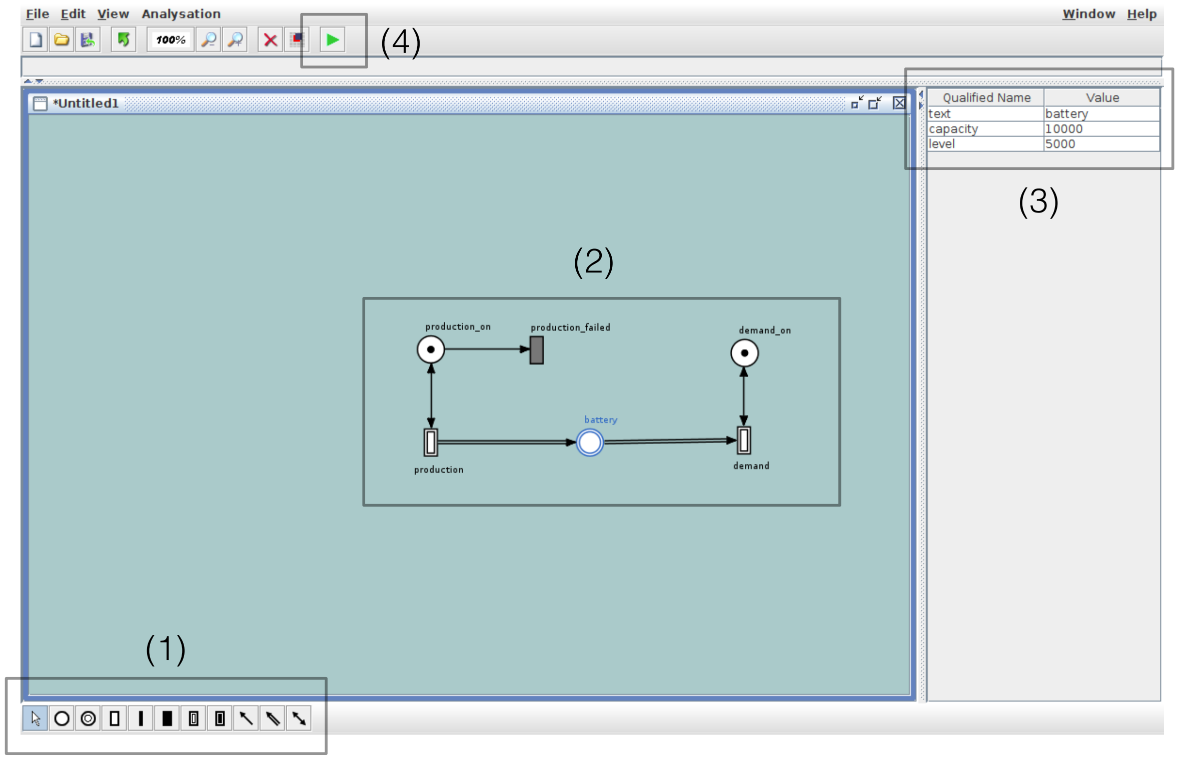
Figure 5.4 shows the main TimeNET window that is used to create a HPnG model. The toolbar **(1)** contains all the HPnG elements implemented in TimeNET. Click-and-drop creates the selected element in **(2)**. By clicking on an element its property window **(3)** is shown. It

```
 public class continuousPlace extends Place {

    public continuousPlace(View parentView, Element element,
                                       Editor netEditor) {
          super(parentView, element, netEditor);
    }

    @Override
    public boolean canConnectTo(Connectable c, String arcType) {
          if (super.canConnectTo(c, arcType)){
                 if (c instanceof generalTransition
                     || c instanceof immediateTransition
                     || c instanceof deterministicTransition) {
                        if (arcType.equals("guardArc"))
                               return true;
                 }
                 if (c instanceof staticContinuousTransition
                     || c instanceof dynamicContinuousTransition
                     || c instanceof fluidTransition) {
                        if (arcType.equals("continuousArc"))
                               return true;
                 }
          }
          return false;
    }
}
```

Figure 5.3. Implementation of a continuous place java class.



Qualified Name	Value
text	battery
capacity	10000
level	5000

Figure 5.4. Using TimeNET to implement a HPnG model.

is used to specify the elements properties. In the example the continuous place *battery* is selected. The properties *capacity* and *level* are exactly the ones implemented in the xsd file. The property *text* is inherited from the general *PlaceType* label.

The editor is extended by a button **(4)** to start the FST. By the use of TimeNET we introduced a model description based on xml-files. Thus when the button **(4)** is pressed a xml-file is created describing the HPnG model. Figure 5.5 shows the xml representation of the model presented in Figure 5.4.

Since the FST did not support the newest analyzation algorithms nor xml-files it had to be updated before it may be used in this research. We briefly introduce the FST, its main functionality and the made extensions in the following section.

5.2 Analyzation and simulation of HPnG models

The actual software used to analyze HPnG models is a console application written in C++, called *HPnGAnalyzer* in the following. It fulfills all requirements for advanced users that are comfortable with HPnGs in general and with the underlying code. A more simple interface, as offered by the FST, is needed that supports the investigation of HPnG models. The HPnGAnalyzer may later be used to perform advanced computations, but the FST can initially be used to investigate a models behavior. Fortunately the FST was first developed by extending the first version of the HPnGAnalyzer. Thus the software architecture of the FST and HPnGAnalyzer do not differ largely and the update focusses on implementing the newest

```
<?xml version="1.0" encoding="UTF-8"?>
<HPnG>
  <places>
    <discretePlace id="production_on" marking="1"/>
    <discretePlace id="demand_on" marking="1"/>
    <continuousPlace capacity="10000" id="battery" level="5000"/>
  </places>
  <transitions>
    <deterministicTransition discTime="10" id="production_failed"
      priority="0" weight="1.0E0"/>
    <fluidTransition id="production" priority="0" rate="17" weight="1.0E0"/>
    <fluidTransition id="demand" priority="0" rate="10" weight="1.0E0"/>
  </transitions>
  <arcs>
    <discreteArc fromNode="production_on" id="0.10" priority="1"
      share="1.0E0" toNode="production_failed" weight="1.0E0"/>
    <continuousArc fromNode="production" id="0.8" priority="1"
      share="1.0E0" toNode="battery" weight="1.0E0"/>
    <continuousArc fromNode="battery" id="0.9" priority="1"
      share="1.0E0" toNode="demand" weight="1.0E0"/>
    <guardArc fromNode="production" id="0.6" isInhibitor="0"
      priority="1" share="1.0E0" toNode="production_on" weight="1.0E0"/>
    <guardArc fromNode="demand" id="0.7" isInhibitor="0" priority="1"
      share="1.0E0" toNode="demand_on" weight="1.0E0"/>
  </arcs>
</HPnG>
```

Figure 5.5. HPnG model xml representation.

analyzation algorithms. Since the software architecture is described in detail in [18] we focus on presenting the FST functionality and the made extensions in the following.

5.2.1 Loading a model

The FST is a C++ window application offering several methods to investigate HPnG models. Initially a *.m-file* is loaded that describes the model under research. In TimeNET the model is saved as an xml-file and thus it is needed to likewise use this xml model representation in the FST to be able to combine both tools later. Furthermore it is advisable to use a standard markup language to facilitate the work with the tools. Thus first an xml-parser has to be implemented that replaces the parsing of a *.m-file* and creates the data structure regarding to the new xml model representation. We used the C++ xml-parser RapidXml 1.13 [32] which is a very lightweight, fast and easy to use library. The modular implementation of the FST tool made it comfortable to replace the *.m-file* parser with the new xml-parser. As presented in Figure 5.6 the latest implementation of the FST now uses the xml model representation instead of the old *.m-file*.

5.2.2 Model checking

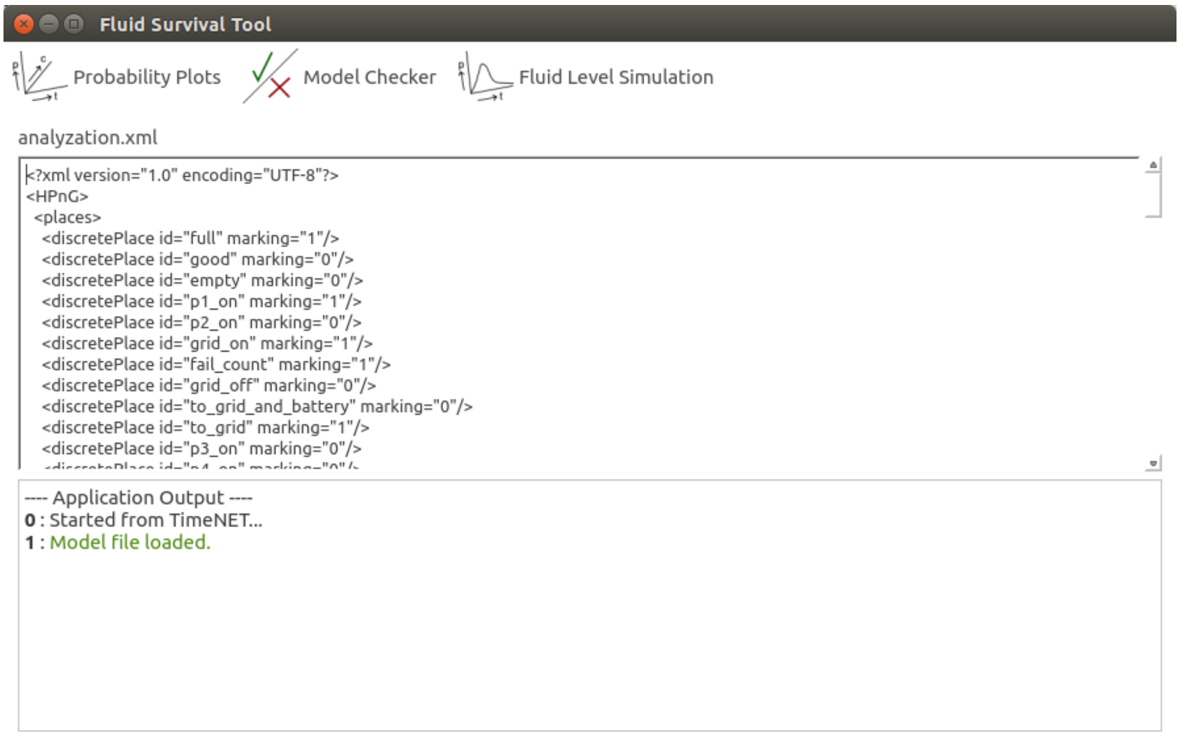
As presented in [18] the FST is a tool that can be used to model check HPnGs based on the STL as introduced in [6]. Inside the *ModelChecking* dialog a user can determine a formula as well as a time to check and thus perform a model checking on HPnG models. As a result he gets weather the current model satisfies the formula or not.

Likewise the FST supports the analyzation of the fluid level of a continuous place. Therefore the user can observe the probability whether the fluid level is at least $x \in \mathbb{R}$. x can be fixed value defined in the GUI, or it can be a parameter with certain start, stop and step values. The computation is done for several points in time. Thus the values are presented as a two dimensional graph, in case of a fixed value x, or a three dimensional graph whenever x is determined to be a parameter. The probability plots are immediately presented by GNUPlot [33], a command line driven graphical utility for all important platforms.

Furthermore the FST supports the investigation of models having more than one general transition through a hybrid form of simulation. Again the fluid level bound x is observed as presented above. One general transition is fixed and for all other transitions a time to fire is chosen according to their random distribution. Thus, instead of one, all general transitions are transformed to deterministic ones. Then the original analyzation algorithms are used to obtain the probabilities. This procedure is repeated several times to get reliable results.

5.2.3 Simulation

To be able to obtain all the results needed for this thesis we need a possibility to get more insights in the evolvement of a HPnG model. It may be helpful to be able to simulate the current fluid level of a continuous place for different points in time. Actually we only support



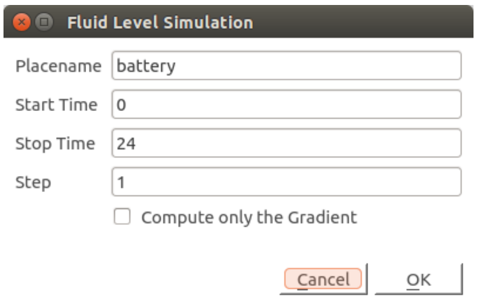
```
<?xml version="1.0" encoding="UTF-8"?>
<HPnG>
 <places>
  <discretePlace id="full" marking="1"/>
  <discretePlace id="good" marking="0"/>
  <discretePlace id="empty" marking="0"/>
  <discretePlace id="p1_on" marking="1"/>
  <discretePlace id="p2_on" marking="0"/>
  <discretePlace id="grid_on" marking="1"/>
  <discretePlace id="fail_count" marking="1"/>
  <discretePlace id="grid_off" marking="0"/>
  <discretePlace id="to_grid_and_battery" marking="0"/>
  <discretePlace id="to_grid" marking="1"/>
  <discretePlace id="p3_on" marking="0"/>
```

```
---- Application Output ----
0 : Started from TimeNET...
1 : Model file loaded.
```

Figure 5.6. The GUI of the fluid survival tool.

the investigation of the fluid level in case no general transition is enabled. This restriction is sufficient for this research purpose and allows to omit the need of several simulation runs which makes it much faster. The current implemented hybrid form of simulation is not suitable to get the fluid level at all points in time. The analyzation is based on the presented STD from Section 4.3.2 and thus work on regions which makes it very elaborate to get the fluid level at all points in time. We implemented a discrete event simulation that makes no use of any analyzation techniques.

As presented in Figure 5.6, the FST is extended by a button *Fluid Level Simulation* to start the simulation. As shown in Figure 5.7 the name of a continuous place is used to identify the place of interest. The start and stop time determines the points in time at which the fluid



Placename	battery
Start Time	0
Stop Time	24
Step	1
☐ Compute only the Gradient	

Figure 5.7. Fluid Level Simulation dialog.

level should be computed. The step size implicitly defines the number of computed values within this interval. It is possible to obtain the absolute fluid level or to compute the gradient by checking the associated checkbox. Once the computation is done again the results are presented directly by Gnuplot as presented in Figure 5.8.

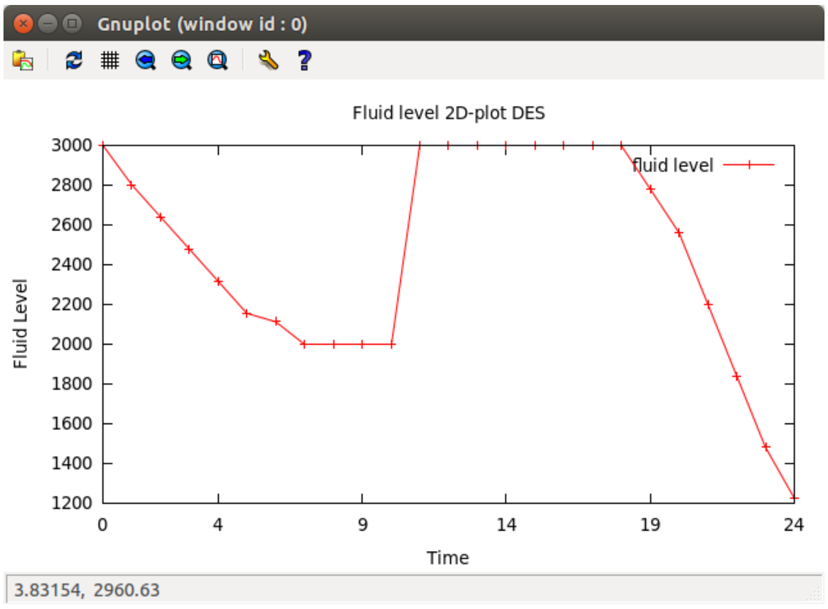


Figure 5.8. The results of a Fluid Level Simulation presented in Gnuplot.

5.3 Combining TimeNET and the FST

We want to provide a tool that both supports the creation and investigation of HPnG models. Thus we have to add the possibility to directly call the FST from TimeNET. The above described development steps made the integration of the FST within TimeNET comparatively easy. As presented above by clicking button (4) as shown in Figure 5.4, a xml-file is produced. At the same time the FST is called from within the TimeNET code as an additional process. The file path to the created xml-file is attached to the FST as a parameter. Therefore the FST directly uses the HPnG model created before and the user immediately can start the investigation. The xml-file created by TimeNET is a temporary file used to be able to combine TimeNET and the FST. Thus this xml-file is the single point of data exchange between these two tools.

We summarize the main steps taken to create and analyze or simulate a HPnG model briefly:

1. Create a model inside TimeNET and hit the button to run the FST in the TimeNET toolbar.

2. The FST is called having the created model as a parameter.

3. Run the analyzation or simulation to do the investigations.

4. Results are presented using GNUPlot.

In case an error occured when running the FST, the process quits and an error code is passed as a return value to TimeNET. This may enable the user to correct possible modeling mistakes and rerun the FST.

5.4 Advanced evaluation

TimeNET in combination with the FST provides sufficient functionality to get a first insight on how a HPnG model behaves and if the implemented model works as expected. However the offered functionality is too much restricted to do all the investigations needed in this research. To implement a GUI that is that much adaptable to support all investigation possibilities, i.e. model checking formulas, parameterization of all model values and computational combination of obtained results, would fairly exceed the scope of this thesis. Therefore the above mentioned HPnGAnalyzer has also to be used to do further analyzation. We then are able to implement the advanced evaluations directly within the code. Thus during the development also the HPnGAnalyzer had to be adapted to work with the new xml model representation.

6 Battery management

We want to investigate the impact of grid-convenient battery charging strategies on self-use, survivability and integrity within this thesis. First, recent battery discharging strategies are presented in Section 6.1. The following sections then cover grid-convenient charging. Section 6.2 presents battery management strategies in smart homes that introduce grid-convenience. Finally the grid-convenient charging of electric vehicles is covered in Section 6.3.

6.1 Battery discharging strategies

In [3] three different battery management strategies were used to investigate the survivability probabilities of smart homes:

Greedy The battery is always discharged for its full capacity. Charging only takes place whenever the local renewable power source is available.

Smart When the grid is available, the battery will be discharged until a certain state of charge to keep a back-up energy. This energy can be used in case of grid failures. Like the *Greedy* strategy, the battery is only charged by the local power source.

Conservative Like in *Smart* strategy, a back-up energy is kept in the battery. If the grid is available, the battery will be charged from the grid up to a second level of charged to keep additional back-up energy.

The *conservative* strategy offers the highest survivability through the additional backup energy. But it likewise utilizes the highest amount of power from the grid. If the client may want to reduce the power used from the grid, for example to reduce the costs, he may want to use the *greedy* strategy. Hence already these original management strategies require a tradeoff among survivability and cost reduction.

The three strategies presented above do not work in a grid-convenient way. Certainly we have to focus on the users requirements since he has to pay for the storage system. But as mentioned in [12] grid-convenience will be an important task towards a more balanced grid. The clients interests may never be neglected, but rather combined with the grid operators interests to get an optimal solution. Thus the impact of grid-convenience on the clients interest are investigated to get a best tradeoff.

6.2 Grid-Convenient battery management in smart homes

In the following we present four different battery charging strategies, as presented in [12], namely (i) direct loading, (ii) delayed loading, (iii) peak shaving and (iv) prediction-based loading.

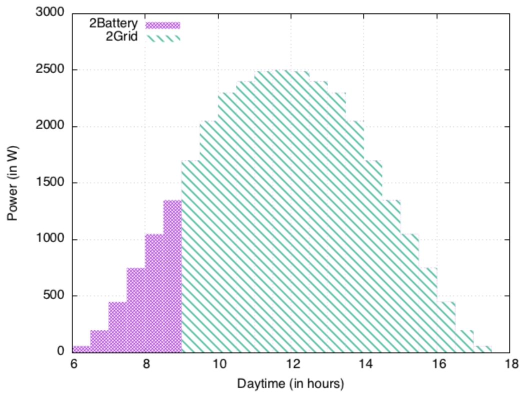


Figure 6.1. Direct Loading

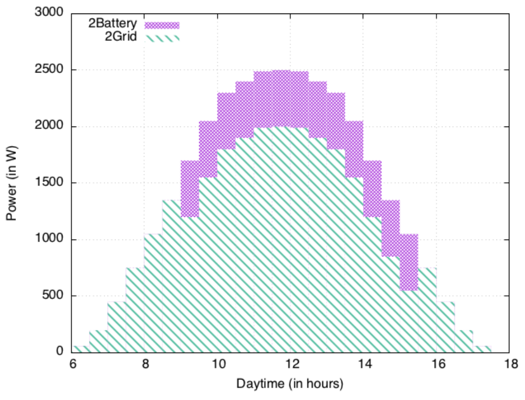


Figure 6.2. Delayed Loading

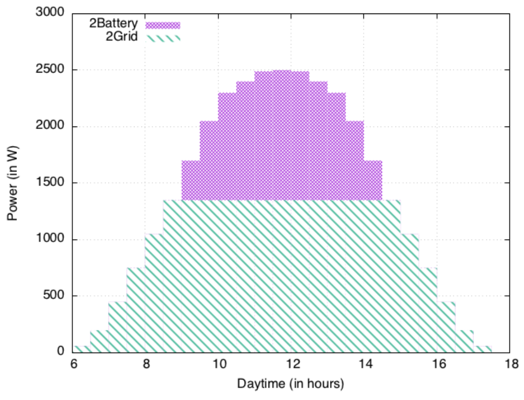


Figure 6.3. Peak Shaving

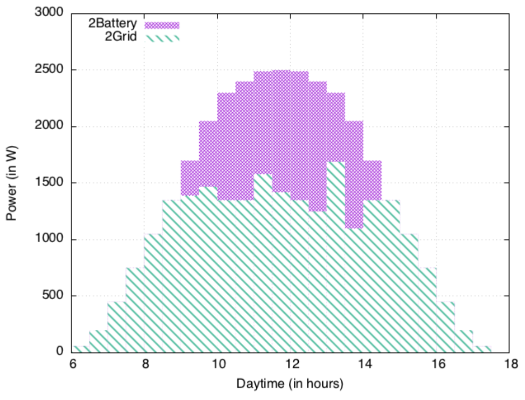


Figure 6.4. Prediction Based

When considering grid-convenient battery management strategies, the focus lies on timing the charging process of the local storage such that the peak supply to the grid is reduced. Whenever possible the local demand should be served from local sources and the battery should always be charged to its maximum capacity during the day to maximize the self-use. In case the production exceeds the demand the battery controller has to decide whether to forward this so-called *overflow* power to the grid or to charge the battery.

Direct Loading charges the battery as soon as overflow is available. When the battery is fully charged, the overflow is completely forwarded to the grid. This strategy is expected to maximize the battery charge during the day and has been used in previous work [3], where the main focus was on survivability.

Under normal operations, the battery is less charged in the morning, when overflow becomes available. As shown in Figure 6.1, *Direct Loading* then first charges the battery before it forwards the complete production peak to the grid. Note that this strategy also leads to an increased feed-in gradient, which goes from zero (while the battery is charged) to the current peak production (as soon as the battery is fully loaded). The other three strategies aim at forwarding less power to the grid during peak production times, which leads to much more grid-convenient schemes. In the following, we investigate in how far this affects the survivability and self-use of the local storage system.

The **Delayed Loading** strategy only stores a certain percentage of the overflow to the battery, after a certain starting time and until it is fully charged. As illustrated in Figure 6.2, this strategy reduces the total peak load that is forwarded to the grid and keeps the gradient much smaller. This strategy highly depends on the chosen loading percentage. It needs to be chosen according to the size of the battery and the prodution (and demand) prediction to ensure that the battery is fully loaded (if possible at all) even during a cloudy day. In case the actual production does not meet the prediction the battery is not fully loaded during a day, even though this might have been possible with better parameters.

Peak Shaving also uses a cut-off value that defines when the battery is charged. All overflow that exceeds the predefined threshold is forwarded to the local storage unit. Hence, the production that is forwarded to the grid will never exceed the cut-off value, resulting in a gradient of zero during peak hours. This is illustrated in Figure 6.3.

As originally presented in [12] the *Peak Shaving* strategy uses a static cut-off value that corresponds to never forwarding more than 60% of the whole PV system performance to the grid, according to the KfW and EEG recommendations. Since this does not take the client's perspective into account, we propose to chose the cut-off value such that it matches current weather predictions and the size of the battery.

In contrast to *Delayed Loading* and *Peak Shaving* the **Prediction Based** strategy does not require pre-defined thresholds, but decides on-the-fly how much energy is forwarded to the grid at the moment. This is visualized in Figure 6.4, where the amount of energy forwarded to the grid may change every time unit, e.g. based on the current state of charge and new predictions. However, this strategy requires a reliable adaptive forecasting system, which is very difficult to realize in the current model. Hence, in the following, we restrict ourselves to the first three strategies.

The strategies presented above introduce grid-convenience. The grid should be stressed as little as possible. Certainly, batteries with higher capacities increase the self-use and thus relieve the grid even more. By the use of electric vehicles the available battery capacity becomes likely such as great, that all of the produced energy will be self-used. The charging may be completed by the grid and hence electric vehicles may be a reason for a higher grid load. We present an approach to charge electriv vehicles grid-conveniently in the following section.

6.3 Grid-Convenient charging of electric vehicles

Electric vehicles provide a very high battery capacity. Whenever the local power production does not suffice to completely charge the EV, the grid is used for charging. The presented strategies aim at delaying the charging to introduce a grid-coordinated charging process. Knowing the total time of charging, a smart grid may decide when to start the charging process on its own. Furthermore the delay prepares the battery for a passive relieve of the grid, i.e. providing free capacities for load balancing or frequency compensation.

In this section we investigate a **just-in-time (JIT) charging** strategy. We therefore assume that the total charging time is known. Whenever the client prepares the car for charging, he determines the total available charging time. For example, in case the client wants to load the car at the working office, the total time of charging may likely be around 9 hours. Preferably the car has to be in the highest possible state of charge when the client returns. In case the charging time does not suffice to completely load the car, the above strategy cannot be used to implement coordinated charging. The charging has to start immediately to reach a high as possible state of charge. Thus we have to focus on charging processes that take a longer time.

As mentioned in [34] and [35] we generally can distinguish between three different qualities of charging stations, that can be used to charge the Nissan Leaf:

- Level 2 (240V) - around 8 hours of charging
- Level 3 (480V) - 3 to 4 hours of charging
- Direct Current Fast Charger (DCFC) - 30 minutes of charging

The availability of DCFCs means that we are able to introduce very fast charging policies.

The JIT charging strategy provides a large battery capacity for a coordinated use. This leads to two main advantages:

1. Delay of charging in case of high loads.
2. A puffer for load balancing and frequency compensation.

Whenever the grid is highly loaded, the knowledge of the total charging time enables the system to delay the charging process. This results in grid-convenience since the charging does not stress the grid whenever it is not needed. Furthermore due to the delay of the charging process, free battery capacity is provided to the grid. This capacity can be used to adjust the voltage level in case of a high loaded grid. Hence JIT charging offers two possibilities for a grid-convenience.

As mentioned in [36] preferably the EV has to be charged between 20% and 80% to extend its battery lifetime. We will not treat this recommendation because of two reasons. First, the upper bound of 80% can only be met by the system when the client specifies it. This means that the system may only load the car just up to 80% when the client wants that. Secondly the lower bound of 20% means that again charging has to start immediately whenever the current state of charge is less than 20%. The battery then is loaded up to 20% and afterwards the JIT charging will take place. Thus only the total available battery capacity for a coordinated use is reduced. Hence it does not influence the results of the JIT charging and thus we do not meet this recommendation in the following. We investigate six different charging strategies regarding to the qualitity of the available charging station:

Greedy Immediately start loading with maximum rate. Assumes that the power source is able to load the car in around 3 hours.

Linear Immediately start loading with a rate to have a completely loaded battery JIT. Mostly this strategy is used to reach convenient charging at home, since the power sources available at home may not be that fast.

Convenient Assumes that a fast power source is available, i.e. the car can completely be charged in at least two hours. Thus the loading can be delayed, but it has to be finished JIT.

Convenient optimistic Works like the convenient strategy but it is expected that the client will likely recur later than determined.

Convenient pessimistic The opposite of the *convenient optimistic* strategy. It is expected that the client will likely recur earlier.

Convenient pessimistic fast Again the client is expected to recur earlier as determined by the total time of charge, but it is assumed that a high-quality power source is available which can charge the car in 1 hour.

The Greedy strategy is the only one that does not aim at JIT charging. Please note that whenever the time to charge does not suffice to completely charge the battery, the Greedy strategy will be used to ensure an as high as possible state of charge.

7 Grid-convenient smart homes

It is particular important that smart homes are continuously supplied with power. A failure of the power supply may cause an unpredictable disaster. In case of a grid failure, batteries should help to survive. Likewise clients want to use batteries to reduce the cost of electrical energy usage. The use of the grid should be as minimal as possible.

In [3] a Hybrid Petri Net (HPnG) model [5] has been presented that allows to analyse the effect of different battery discharging strategies on the survivability of a smart home. We present this abstracted smart home model in Section 7.1 and enhance it with a detailed description of the usage pattern model in Section 7.2. A extended smart home model then is presented in Section 7.3. The parameters choices for this extended battery management are presented in Section 7.4. Finally Section 7.5 summarizes the computed results for grid-convenience, self-use and survivability. Section 7.6 discusses the computed results and concludes this chapter.

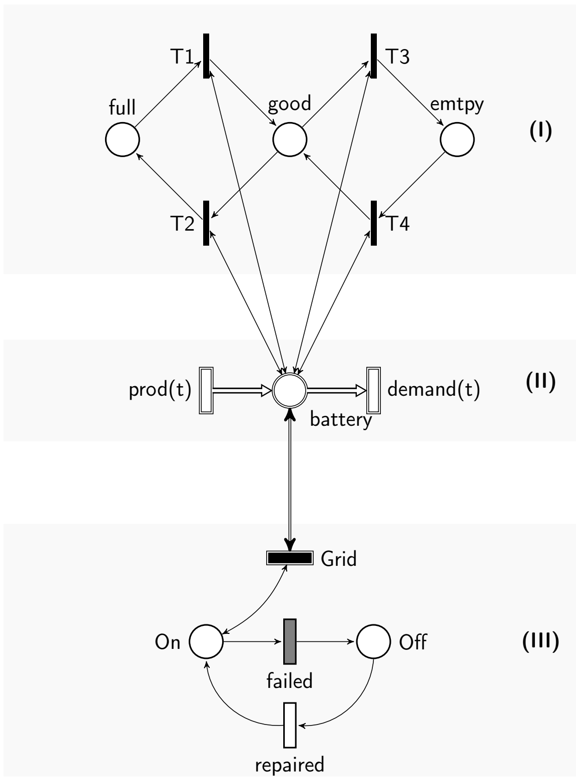


Figure 7.1. Smart house HPnG model.[3]

7.1 HPnG model

Figure 7.1 shows an abstracted smart home model. It consists of three different parts: (I) the battery management unit, (II) the battery model along with the production and demand and (III) the grid model.

The smart home and the battery are represented by a single continuous place. It is connected to the *time-dependent* transitions $prod(t)$ and $demand(t)$, which represent the deterministic production and demand profiles during the day.

The *Battery Management unit* controls the flow of power between the local generation, and the house, depending on the battery management strategy used. The model distinguishes between three states of charge. The battery can either be *full*, *good* or *empty*, where the state *empty* can be interpreted relative to the overall capacity of the battery B. The transitions T_i for $i \in \{1, 2, 3, 4\}$ coordinate the change of state via test arcs that enable the firing of transition T_i according to some threshold that is compared to the available capacity of the battery.

Energy is imported from the grid to the house when there is not enough local energy available. The interaction between the battery and the grid is represented by a bidirectional continuous arc. However, this interaction is only possible when the grid is operational. When the grid fails (modeled by the deterministic transition t_{failed} at time a) the token is moved to place *Off* and the house is practically isolated from the grid. This allows to analyze the impact of different times of failure on the survivability. The grid recovers from its failure according to a stochastic repair distribution that can be chosen freely. Note that we assume that if in case of a grid outage during peak production times the battery is completely charged, the local production is reduced to match the current demand.

7.2 Production and demand setting

Similar to [3], we consider a scenario in which the yearly production is equal to the yearly demand; both have been set to 3 kWh. The used production profile generally distinguishs between days of high and low production, as well as between summer and winter. This leads to four production profiles. The demand is separated in summer and winter and thus two patterns are presented. We want to investigate grid-convenience and therefore focus on times of high overflow. Hence in the following we use the summer high production and the summer demand profile.

The production profile for the PV-panels, as presented in Figure 7.2, is obtained from the internet tool PVWatts by the National Renewable Energy Laboratory [37]. This tool provides a full year of PV-production based on the characteristics of the PV-installation, such as size, orientation and system losses. Actual data from weather stations around the world is used to generate a yearly production profile, with 1 hour granularity. We have used the weather data from the Amsterdam station and Figure 7.2 shows the resulting production for a sunny and for a cloudy summer day. The values are approximated according to [3].

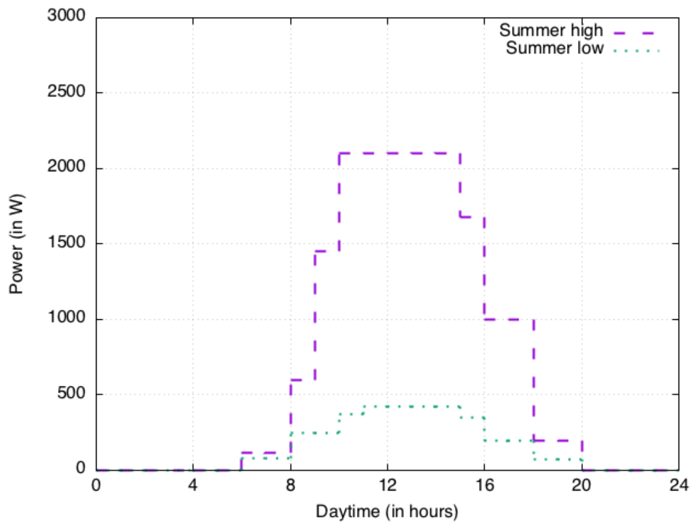


Figure 7.2. Approximation of the production profile for a summer day (sunny and cloudy)

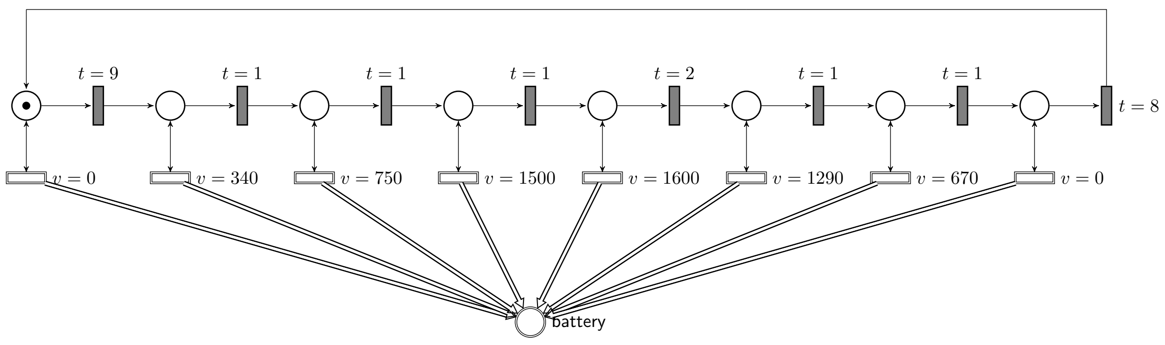


Figure 7.3. Example smart house production model for *winter high* profile.

Due to the approximation we have a finite number of value changes. For at least one hour, the production remains constantly the same. As shown in Figure 7.3 such a pattern can be implemented as a sequence of deterministic transitions. Each of these deterministic transitions represents the moment in time whenever the production value changes. If a certain time bound is reached, the discrete marking changes which means that another static continuous transitions becomes enabled. Figure 7.3 presents the detailed production model for the *production low* profile.

The approximated demand profile is presented in Figure 7.4. For the demand, we use a standard profile provided by EDSN [38]. EDSN provides different standard profiles for the Dutch market. We use the E1a profile, for a connection smaller than $3 \times 25A$ with a single counter. The profile gives the demand for a full year at a 15 minute granularity. Figure 7.4 approximates the daily demand for an average summer and for an average winter day.

We now extend the presented smart home model in the following section. This adaption may lead to a more realistic model and enables to implement grid-convenient battery management strategies.

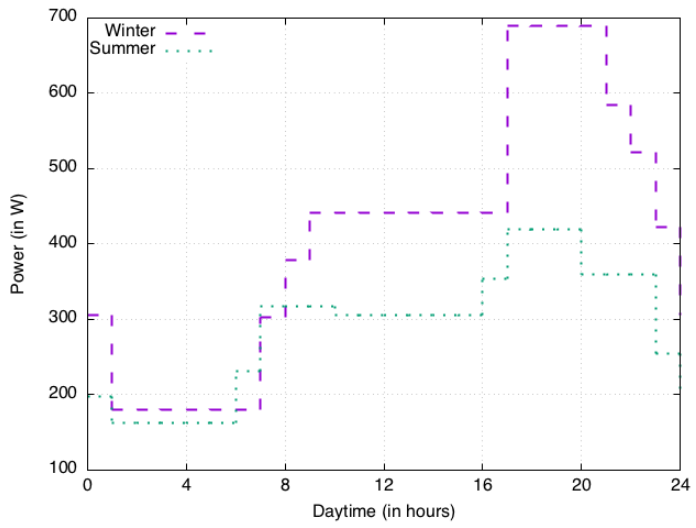


Figure 7.4. Approximation of the EDSN demand profiles for a summer and winter day.

7.3 Model implementation

We extend the smart home model in the following ways: (i) the battery is explicitly modeled and separated from the smart home. (ii) This allows to model the loss that results from storing energy in the battery. (iii) The control part is extended in order to allow an implementation of the grid-convenient battery management strategies.

Figure 7.5 presents the extended smart home model. The smart home is represented by a continuous place with capacity zero, since as in any other (micro) grid, the sum of the energy provided and the energy consumed should amount to zero. The house is connected to the *time-dependent* transitions $prod(t)$ and $demand(t)$. Additionally, the house is connected to the battery, which is modeled as a continuous place with overall capacity B, and the grid, which is modeled as a place with infinite capacity.

Whenever the battery is charged via transition $t_{battery}$, additional energy is taken from the house through transition t_{losses}, modeling the fact that charging and discharging the battery cannot be done without loss. For simplicity, we have chosen to attribute this loss to the charging process, only.

The *Battery Management unit* (BM) still remains the same but the strategies *Delayed Loading* and *Peak Shaving* require an additional control instance, i.e., the Extended Battery Model (EBM). It ensures that the battery is only loaded if the requirements specified by the battery management strategy are met. The battery can only be charged if there is a token in place $2battery$ (realized via a test arc) and in case there is no token in place $full$ (realized by an inhibitor arc). Note that the amount of power that is stored in the battery per time unit is managed by the dynamic transition $t_{battery}$. Depending on the battery management strategy this transition either forwards a predefined percentage of the overflow or the overflow that exceeds a certain threshold to the battery.

Likewise the bidirectional connection to the grid remains the same.

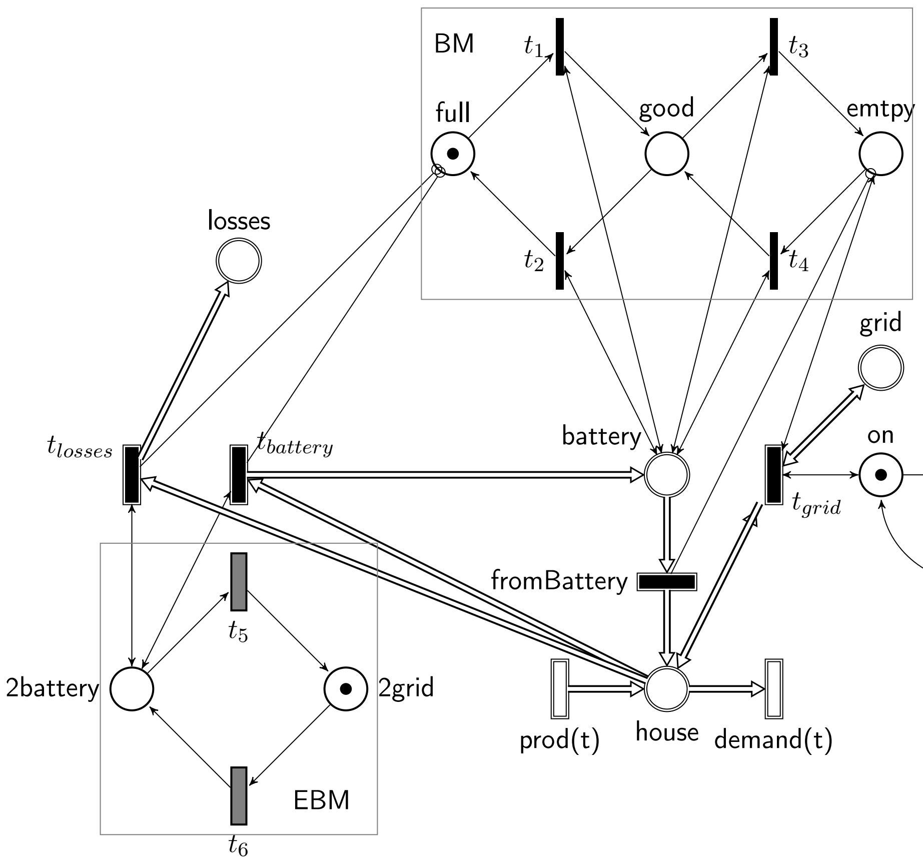


Figure 7.5. HPnG model of a smart house with delayed loading strategy.

For finite time bounds the model can be analyzed with efficient techniques for Hybrid Petri Nets that allow one stochastic variable in the model. This stochastic variable is used to model the time the grid needs to become operational again after an outage.

7.4 EBM parameter choices

Battery discharging always follows the *greedy* strategy (c.f. [3]). The authors show that *Greedy* maximizes the *self-use* and drains the battery whenever the demand exceeds the production.

Whenever *Delayed Loading* is enabled as charging strategy, the battery may only be charged with a certain percentage of the overflow per time interval. From Figure 7.2 we see that the production rises considerably between 8:00 and 10:00 hours. We decide to start loading the battery at 9:00 hours and consider different loading percentages and investigate their impact on the defined measures of interest. Hence, transition t_6 in Figure 7.5 fires at 9:00 hours. For modeling convenience, transition t_5 defines an end time for the battery charging process. Note that this is not needed in practice, since the charging process stops automatically, when the battery is fully loaded or when the local production ends. The loading percentage is implemented by the dynamic transition $t_{battery}$.

For *Peak Shaving* we have to define a threshold, such that the overflow that exceeds said value is stored in the battery. We do some simple pre-computations to ensure that the battery can be fully loaded for the current production and demand predictions. For a know production pattern, as e.g., in Figure 6.3 this corresponds to choosing the threshold such that the size of the purple region corresponds to the size of the battery.[1] For the scenario at hand, this computation results in a threshold of $1100W$. For illustration purposes, we also consider the thresholds $950W$ and $1150W$ as cut-off values. Note that, for a any other production pattern the thresholds have to be recomputed. To implement this strategy the dynamic transition $t_{battery}$ is parameterized in a way that it forwards the overflow that exceeds the predefined threshold to the battery.

7.5 Results

This section presents and compares the measures for the three different battery charging strategies *Direct Loading*, *Delayed Loading* and *Peak Shaving*. Section 7.5.1 discusses grid-convenience, Section 7.5.2 considers the resulting self-use and Section 7.5.3 compares the survivability probabilities.

7.5.1 Grid-convenience

We have to compute the amount of power that is forwarded to the grid to investigate the grid-convenience. It is the aim to reduce the supply peak and have as low as possible gradients. Furthermore the battery must be completely charged as soon as no overflow is available anymore.

For *Delayed Loading* we consider several loading percentages, but for visibility only depict the results for loading percentages of 75%, 50%, 25% and 31% in Figure 7.6 together with the values for *Direct Loading*. Figure 7.7 shows the state of charge of the battery during one day for the same loading percentages, as well as for *Direct Loading*. We see that the latter only starts to forward power to the grid at 12:00 hours. Before, the locally produced energy is used to charge the battery. The resulting gradient at 12:00 hours is very steep, going from $0W$ to $1516W$.

When *Delayed Loading* is enabled, the overflow is first forwarded to the grid and only when time $t = 9$ is reached, a fraction of the overflow is used to charge the battery. For high loading percentages (e.g., 75%) the battery is fully loaded at 14:00 hours, still resulting in a steep gradient of power forwarded to the grid. While for lower loading percentages, i.e., 25% and 31%, the gradient is reduced, the battery is no longer fully charged during the day in all cases. Through several experiments a loading percentage of 31% has been shown to fully charge the battery for the given production and demand profile, while forwarding power to the grid in a more balanced way.

The results of similar computations are shown in Figure 7.8 and 7.9 for different cut-off

[1]Note that this kind of pre-computation is also possible for the Delayed Loading strategy, however, it is less obvious.

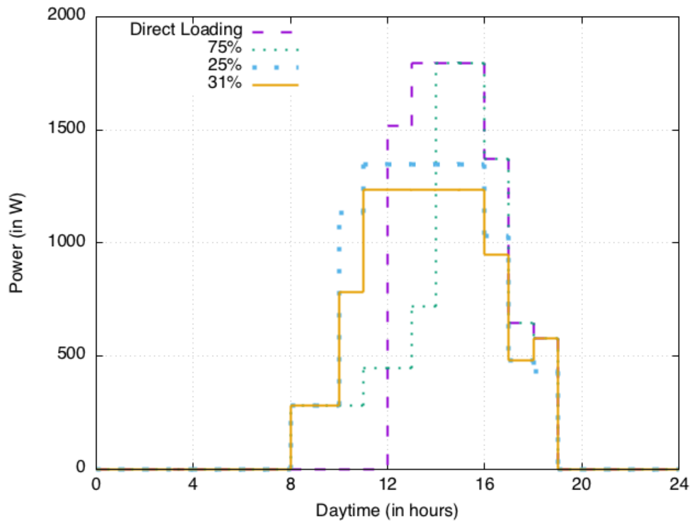


Figure 7.6. Power forwarded to the grid during one day with the *Delayed Loading* strategy.

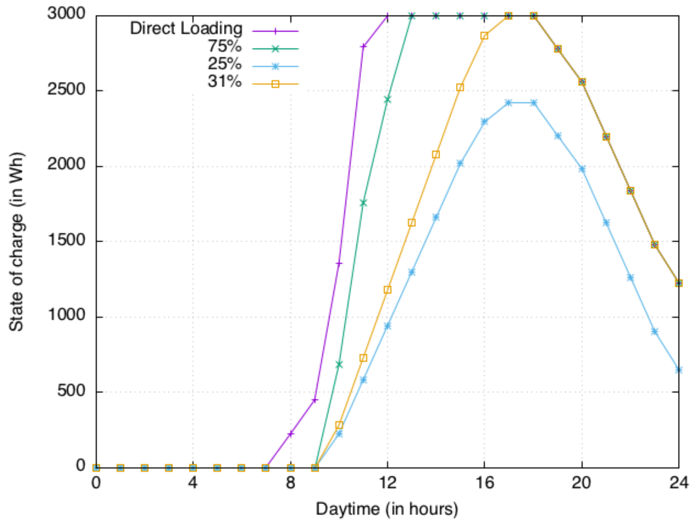


Figure 7.7. State of charge of the battery during one day with the *Delayed Loading* strategy.

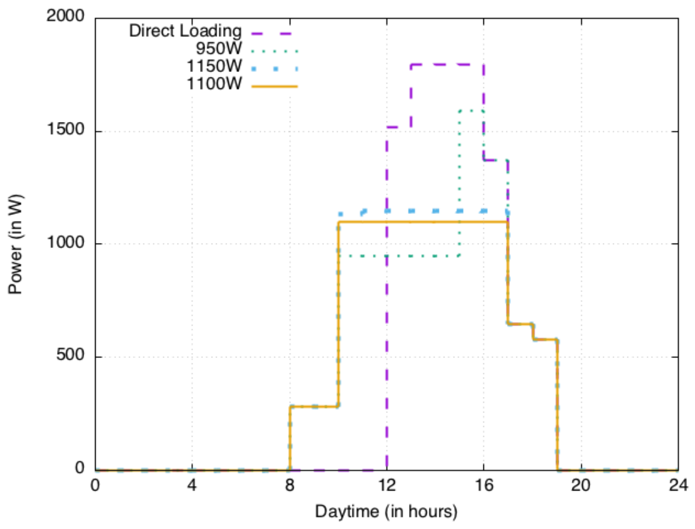


Figure 7.8. Power forwarded to the grid during one day with the *Peak Shaving* strategy.

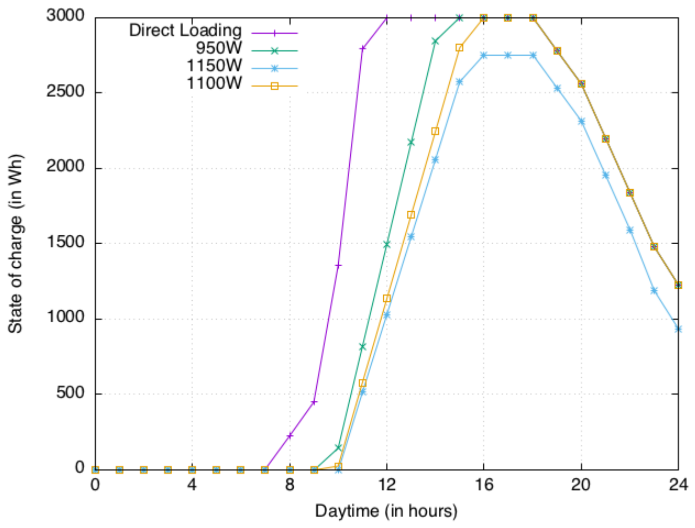


Figure 7.9. State of charge of the battery during one day with the *Peak Shaving* strategy.

Strategy	Max Rate (W)	Max Gradient (W)
Direct Loading	1820	1516
Delayed Loading	1158	580
Peak Shaving	1100	818

Table 7.1. Maximum feed-in rate and gradient per strategy.

values. For comparison we repeat the results for *Direct Loading* in these Figures, as well. The experiments show that for the pre-computed value of $1000W$ the battery is fully charged during the day and the amount of power forwarded to the grid stays just a little over $1000W$. When the threshold is a slightly lower, e.g. $950W$ the battery is completely charged earlier, which however leads to a considerable increase in the power forwarded to the grid at 15 hours. For a slightly larger threshold of $1150W$ the battery cannot be fully charged during the day, while the amount of power forwarded to the grid is slightly larger as for a threshold of $1100W$.

As presented in Figures 7.7 and 7.9 the state of charge decreases again after no overflow is available anymore, since the battery then is used to power the house. However, the state of charge after one day is not the same as the initial state of charge, since the production on such a sunny day exceeds the demand considerably. We do not consider computations for multiple days, since the number of possible combinations would exceed the scope of this thesis. Instead, we aim at showing the potential of the different battery charging strategies in an exemplary setting.

The above shows that the strategies *Delayed Loading* and *Peak Shaving* are both able to improve the grid-convenience of a local storage system. However *Direct Loading* increases the stress-level of the grid, because it adds a very high gradient during peak production hours. It is particular important to choose an appropriate loading percentage for *Delayed Loading* and a cut-off value for *Peak Shaving*, to ensure that the battery is fully loaded during the day.

For the current setting *Delayed Loading* with 31% loading percentage and *Peak Shaving* with a threshold of $1100W$ perform very well. Comparing the resulting curves from Figures 7.6 and 7.8, we see that with *Delayed Loading* the peak forwarding rate to the grid is reached in several small steps, while with *Peak Shaving* the peak forwarding rate is a little lower and constant for a longer while. This nicely reflects that the latter strategy does not forward any fluctuations, that occur above the chosen cut-off value, to the grid.

Figures 7.10 - 7.12 illustrate the use of the overflow between the grid and the battery, approximated in intervals of one hour. Note that losses that occur from charging the battery are shown separately from the battery. *Direct Loading* (c.f. Figure 7.10) does not reduce the peak forwarding rate to the grid. The maximum forwarding rate is $1820W$ with a maximum gradient of $1516W$. The strategies *Delayed Loading* and *Peak Shaving* however charge the battery during production peaks. This reduces the maximum forwarding rate to the grid as well as the maximum occurring gradient, as shown in Table 7.1. *Delayed Loading* results in a slightly higher feed-in rate during peak production, however the resulting gradient is considerably lower than with *Peak Shaving*. Hence we conclude, that for the current setting

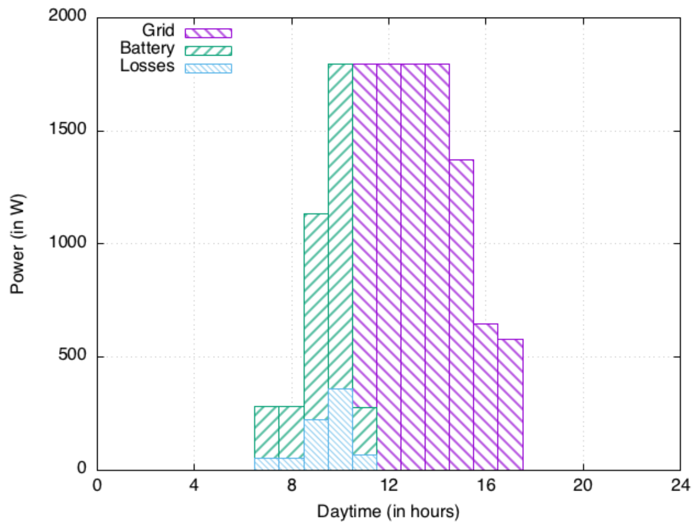


Figure 7.10. Distribution of the overflow during one day with *Direct Loading*.

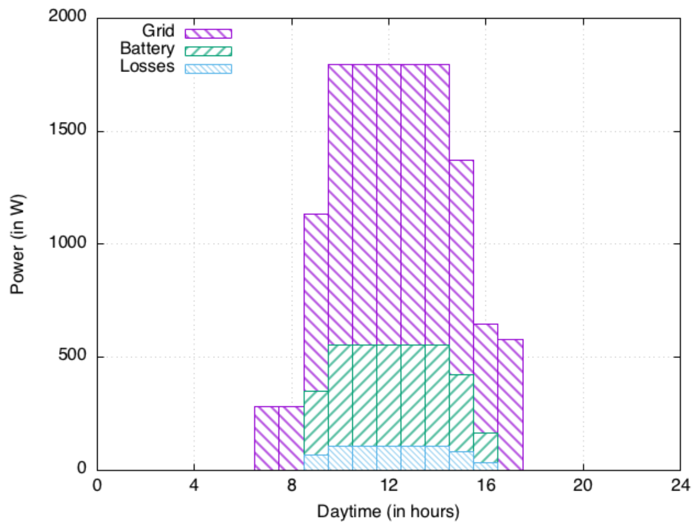


Figure 7.11. Distribution of the overflow during one day with *Delayed Loading* with 31%.

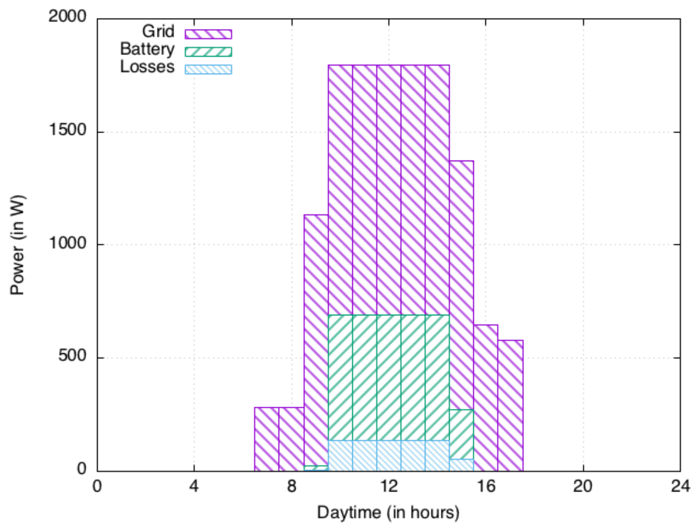


Figure 7.12. Distribution of the overflow during one day with *Peak Shaving* with $1100W$.

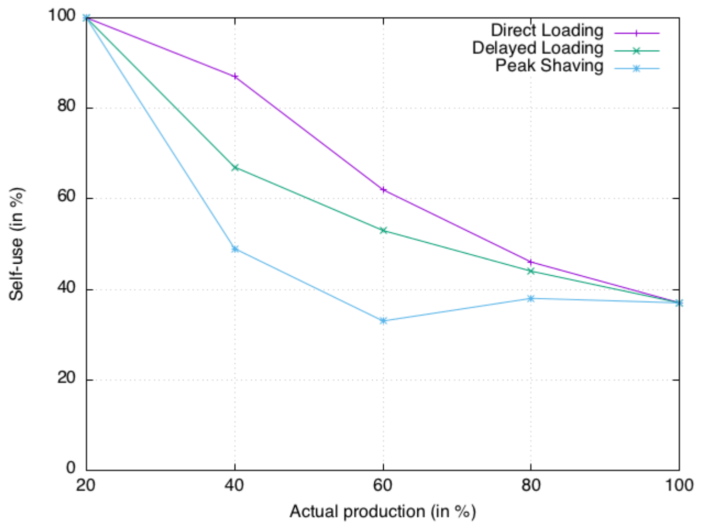


Figure 7.13. Self-use realized with different strategies, depending on the realized production.

Delayed Loading with 31% results in the most grid-convenient local storage system.

These results also show that *Delayed Loading* and *Peak Shaving* both fulfill the requirement of peak supply reduction, i.e. at most 60% of the nominal power of the local production is supplied to the grid. For the current setting, this means that the maximum feed-in rate should not exceed $1800W$. Note that *Direct Loading* does not fulfill this requirement.

7.5.2 Self-use

We would like to stress that the above results are only valid for a specific combination of production and demand profiles. The pre-computations, that can be used to identify appropriate parameters for the more grid-convenient strategies, highly rely on the accuracy of the predicted profiles. Especially, when the production stays below the predictions, this can have severe consequences for the home owner. Hence, this section takes into account the quality of the predictions, when measuring the client-based measures of interest, i.e., the self-use and the survivability.

Figure 7.13 illustrates the amount of self-consumed and locally produced energy as a proportion of the whole amount of locally generated power, for the three discussed battery charging strategies. On the x-axis we depict how much of the predicted production is actually realized. The values for 100% correspond to the results shown in the previous section and the lower percentages model that the weather predictions are not met, e.g. because it is more cloudy than predicted. Note that we only take into account the quality of the production predictions, while we keep the demand fixed, since it is more controllable by the home owner than the production.

A very low actual production of 20% results in a self-use of 100% in all three strategies, because the production does not exceed the demand and hence no overflow is charged to the battery but all production is consumed by the smart home. Thus, the chosen strategy does

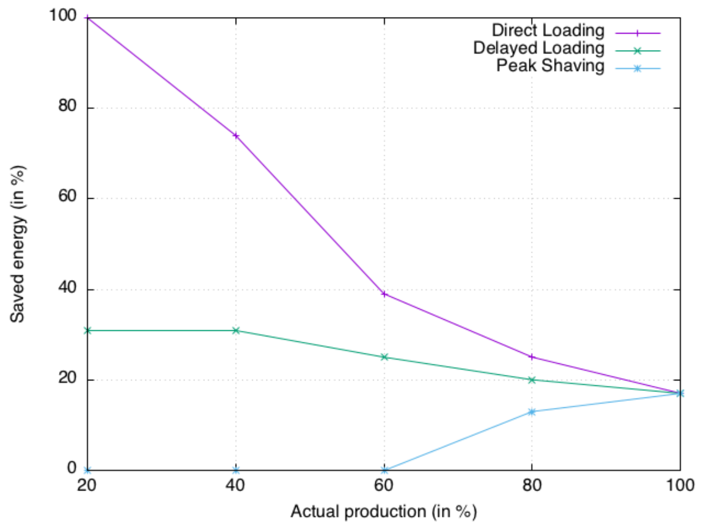


Figure 7.14. Energy charged to the battery with different strategies, depending on the realized production.

not have any impact on this measure. Also in case of 100% production the choice of strategy does not influence the self-use, since the battery is fully charged during the day.

For all production values in between, *Delayed Loading* and *Peak Shaving* result in a lower self-use than *Direct Loading*, because the battery is not completely charged. Figure 7.14 presents the fraction of overflow charged to the battery for all three strategies. We see that the grid-convenient strategies perform much worse than *Direct Loading*, when the predicted production is not realized. *Peak Shaving* performs very poorly due to the absolute cut-off value. *Delayed Loading* is always able to charge to battery to the certain percentage.

We conclude that Peak Shaving is much more dependent on accurate weather and demand predictions. We see that for a realized production of less than 40% *Delayed Loading* is able to charge the battery to the pre-specified percentage of 31%, while *Peak Shaving* does not charge the battery at all for a realized production that is less than 60%. Note that for a higher realized production the percentage of saved energy decreases, depending on the overall available overflow. Even when the battery is completely charged, we are not able to realize a higher percentage, as this value is limited by the size of the battery.

7.5.3 Survivability

The state of charge of the battery also has an impact on the **survivability** of the house. For model checking the survivability as specified in Section 3.3 we define the property `power_available` such that it holds whenever there is charge in the battery or the production at least meets the demand. In all other cases, we do not consider the house to be powered.

Figures 7.15, 7.16 and 7.17 present the resulting survivability for the different charging strategies for two consecutive days. Recall that the survivability provides the probability that

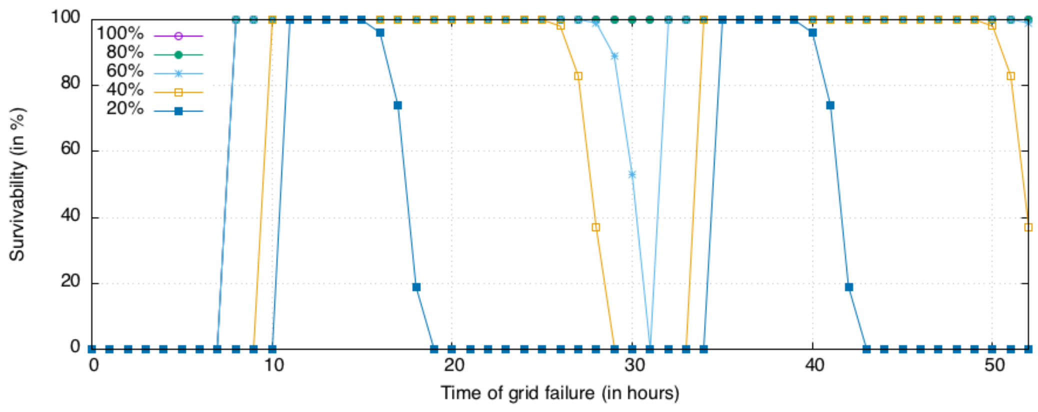


Figure 7.15. Survivability probability for *Direct Loading* for different failure times and prediction qualities.

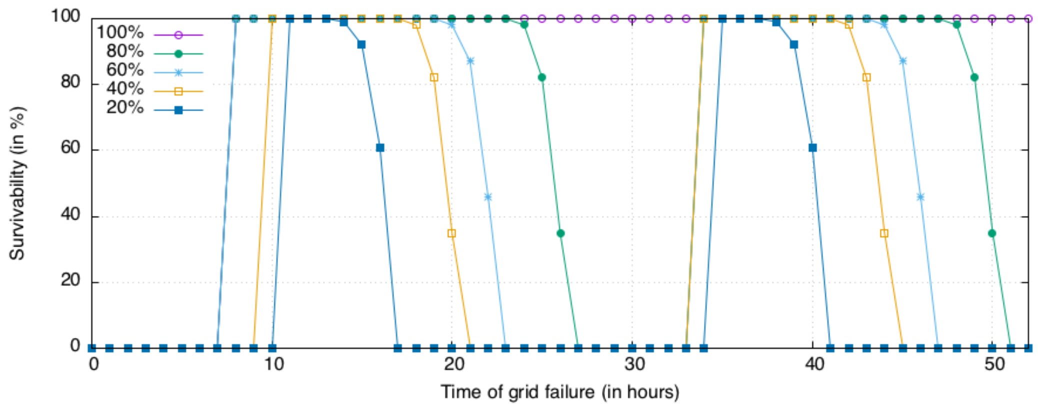


Figure 7.16. Survivability probability for *Delayed Loading* with 31% for different failure times and prediction qualities.

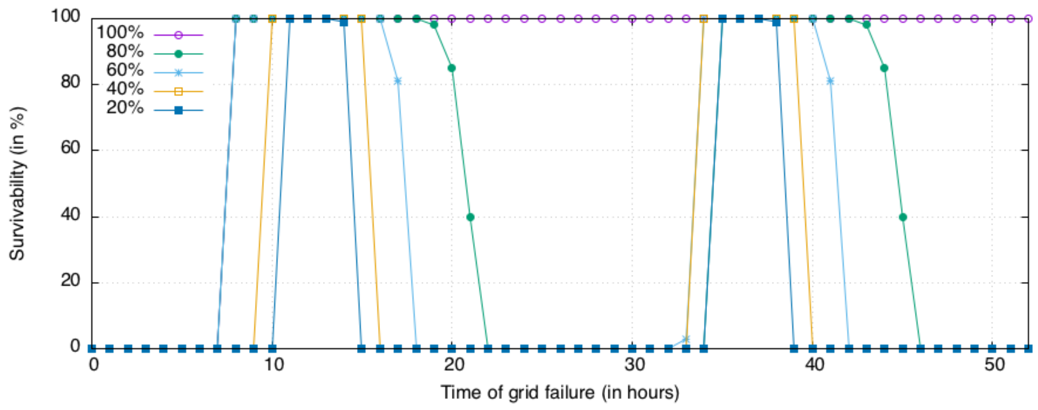


Figure 7.17. Survivability probability for *Peak Shaving* with $1100W$ for different failure times and prediction qualities.

the house can continuously be powered from local storage in case of a grid outage, until the grid returns. The time of failure is depicted on the x-axis and curves correspond to different prediction qualities.

Before 7:00 hours the survivability is 0% for all strategies. This is due to the fact that we start with an empty battery and in order to be survivable, the production at least needs to match the demand. In case at least 80% of the production is realized, *Direct Loading* provides a survivability of 100% for all failure times after 7:00 hours. For lower production values the survivability starts slightly later. However, during the first day the battery will be charged such that there is still charge available to power the house at the next morning. When only 60% of the production is realized, the survivability drops to zero for a very short period during the early hours of the next morning. For even lower realizations the resulting survivabililty gap becomes considerably large.

The results for *Delayed Loading* and *Peak Shaving* indicate that only a realization of 100% of the production is able to ensure a survivability of 100% for all failure times after 7 hours. For *Delayed Loading* a realization of 80% of the production, results in a gap of six hours with zero survivability between 3 and 9 hours of the second day. For a realization of 60% this gap is as large as ten hours.

Comparing these values to *Direct Loading*, we see that 80% of the predicted production still results in 100% survivability and 60% only leads to a very small interval with a decreased survivability. We conclude that *Delayed Loading* is considerably less robust w.r.t. to the prediction quality than *Direct Loading.* As can be seen in Figure 7.17, *Peak Shaving* is even less robust w.r.t. the realized production. Again, 100% production leads to 100% survivability after the 9 hours. A production of 80%, however, results in a gap of 11 hours with zero survivability, which is not acceptable for the owner of a smart home. Comparing the performance of *Delayed Loading* and *Peak Shaving* w.r.t. the survivability, the first performs much better, when the production stays below the predictions, due to the fact that the charging process is distributed more evenly.

7.6 Discussion

We presented an extended Hybrid Petri net model that is able to implement grid-convenient battery managment strategies, like *Delayed Loading* and *Peak Shaving.* The resulting amount of power forwarded, the corresponding maximum gradient and the state-of-charge of the battery for these strategies to the grid is compared with the most straight-forward strategy, i.e., *Direct Loading.*

We have seen that *Direct Loading* results in very large maximum gradients and maximum feed-in rates, which exceed the requirement of peak supply reduction stated by the German KfW [19]. In contrast, the strategies *Delayed Loading* and *Peak Shaving* do fulfill this requirement and hence employing these strategies would make the storage eligible for subsidies. Both strategies lower the gradient of forwarded power by distributing the charging process of the battery over a longer period of time. This emphasizes the relevance of the presented work, as more grid-convenient strategies are necessary if local storage systems are expected to help balancing the grid.

We have also investigated the impact of said strategies on customer requirements, i.e., self-use and survivability. We have seen that *Delayed Loading* and *Peak Shaving* are not robust w.r.t. the quality of production predictions. Both measures of interest suffer considerably, when the realized production does not meet the predictions. This is due to the fact that the cut-off value and the loading percentage need to be precomputed based on predictions. Note that *Peak Shaving* performs considerably worse than *Delayed Loading* in the case of bad predictions. The latter always charges the battery at least to a certain extent, while the first may not charge the battery at all if the realized production stays below the cut-off value.

In general, we can state that *Direct Loading* focuses only on the consumers needs and grid-convenience is neglected. Hence, *Direct Loading* results in the best self-consumption and survivability, independently of the quality of the predictions. On the other hand, the implementation of grid-convenient strategies does not result in a disadvantage for the customers, as long as accurate predictions are available.

One way to improve the dependency on the accuracy of the prediction would be to implement the grid-convenient strategies in a more adaptive way. This would require the system to regularly check whether the realized production meets the predictions and to update the parameters of the strategies, accordingly. However, this cannot be easily included in the Hybrid Petri net model presented in this paper. Future work will investigate the possibilities of combining a learning strategy for charging parameters with a system model that also includes stochastic failure and repair characteristics.

8 Grid-convenient charging of electric vehicles

We presented a smart home model used to assess grid-convenient battery management strategies. Electric vehicles offer additional battery storage to a smart home, which may be grid-convenient. But according to [39], the general charging process of EVs is expected to increase the grid's load. In this section we present a HPnG model used to investigate grid-convenient charging strategies for electric vehicles. We focus on strategies that enable a coordinated use, which may lead to a more balanced grid according to [39]. We encapsulate the current model from the smart home, since the general transition is needed to model the clients recurrence.

The presented charging strategies from Section 6.3 aim at a just-in-time charging. The battery is completely loaded as soon as no overflow is available anymore. In the following we will investigate in how far a just-in-time strategy enables a coordinated use and influences the *integrity* of the charging process.

This chapter is organized as follows. Section 8.1 presents the meaning of coordinated charging. We then present the used HPnG Model in Section 8.2. The parameters used to specify the random distribution and the charging time are presented in Section 8.3. The results are presented in Section 8.4 and finally discussed in Section 8.5.

8.1 Coordinated charging

As mentioned in [39] coordinated charging may help to relieve the grid. It enables a smart grid to (i) determine the start of charging a EV and (ii) use provided battery capacity as a backup for load balancing and frequency compensation. Thus the investigated strategies have to delay the charging process. Note that due to the maximal charging rate, the latest time to start charging the EV is determined by the used strategy.

We value the grid-convenience of a certain strategy by the size of battery capacity that is provided to the grid for a coordinated use. Let c be the total battery capacity and c_i be the state of charge at time i. t is the maximal possible charging time. The average available battery capacity thus can be computed by:

$$1 - \sum_{i=0}^{t} \frac{c_i}{c}$$

The greater this value, the greater the possibility of a coordinated use. We investigate in how far a great available battery capacity influences the integrity of the charging process.

8.2 Model of the charging system

The HPnG model consists of three parts, i.e. (i) the battery management unit, (ii) the battery and (iii) the recurrence model. To illustrate the use, we consider the example of charging the EV at work. The model represents the charging after the client arrived at work and finished to prepare the car for charging.

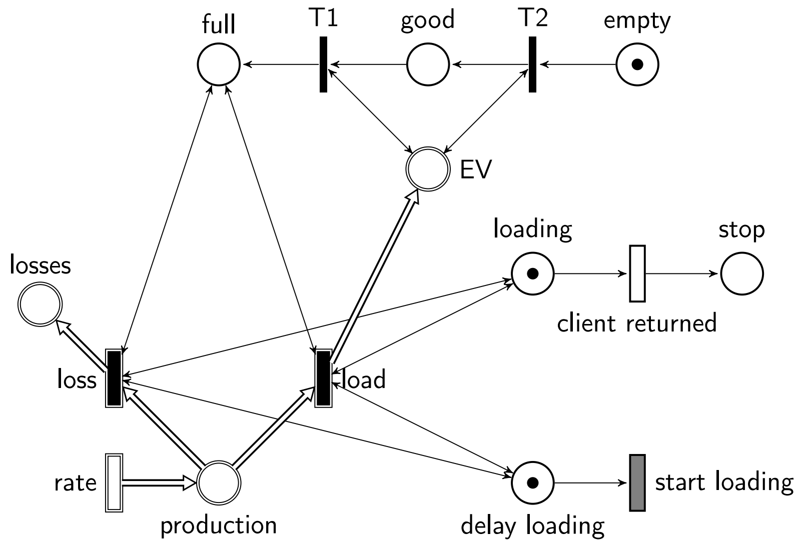


Figure 8.1. Model of charging a electric vehicle.

The model is presented in Figure 8.1. The battery management unit is located at the top of the model and consists of the three discrete places *full*, *good* and *empty*. The unit works as presented in Section 7.1, with the restriction that we do not consider discharging. Hence we omitted all arcs and transitions to move token from *full* to good and from *good* to *empty* respectively.

The continous transition *rate* represents the overall rate r used to charge the battery. The dynamic continuous transitions *load* and *loss* depend on transition *rate*. They charge the battery as soon it is not full and available for loading, i.e. the client did not returned. These conditions are stated by four guard arcs. Furthermore the place *delay loading* must not contain a token, in order to start the charging process. The deterministic transition *start loading* is used to delay the loading process for a certain time t_d. Again two guard arcs are used to ensure that condition.

The clients return is modeled by a general transition *client returned*. We use a folded normal distribution with mean value μ, i.e. the expected recurrence time, and deviation σ. The charging immediately stops as soons as the client returns. Charging can only take place as long as the discrete place *loading* contains a token. Informally spoken, it can not occur that the client waits at the car until it is completely charged. This states that the STL formula presented in Section 3.4 is suitable to model check the *integrity*.

8.3 Parameter setting

As mentioned in the previous section, the model is paramaterized several times. These parameters are used to introduce the different charging strategies and times of the clients recurrence.

As mentioned before, we consider the use of a Nissan Leaf, thus the battery size is fixed at 27kWh. Different battery capacities may influence the total time of charging, but not the integrity probabilities in general.

The charging rate r is based on the different recent available charging stations as presented in Section 6.3. The *Linear* strategy assumes a low performing station, having a rate of $r = 3,75$kW/h. The *Greedy* and *Convenient* strategies use a better performing station which leads to a rate of $r = 11,25$kW/h. The influences of a high performing station are investigated by the *Convenient fast* strategy having a rate of $33,75$kW/h.

As an example of a real charing process, we assess the charging at work. This means that the client is not expected to recur before 9 hours. Hence we use a mean value of $\mu = 9$ to fix the folded normal distribution. The influence of imprecise recurrence times is investigated by parameterizing the deviation σ. We incorporate the values $\sigma = \{0.25, 0.5, 0.75, 1, 1.25, 1.5\}$ hours.

As we focus on just-in-time strategies, the car must be charged completely after 9 hours. The *Greedy* and *Linear* strategy directly start loading and thus the delay time $t_d = 0$. The delay time for the convenient strategies is implicitly determined by the used rate r. The *Convenient* and *Convenient pessimistic fast* strategy start charging at $t_d = 6$. Assuming an earlier respectively a later recurrence, the strategy *Convenient pessimistic* uses $t_d = 5$ and *Convenient optimistic* $t = 7$. Note that we only prepare a charging strategy that enables a coordinated use. We do not allow the grid to start the charging at a arbitraty point in time, but always treat the latest point in time the charging must start.

8.4 Results

This section compares the measures for the just-in-time charging strategies with the *Greedy* charging. We first check whether the strategies work as expected and therefore recall the evolution of the state of charge for the different charging strategies.

Figure 8.2 shows the state of charge, represented on the y-axis, for increasing time on the x-axis. Both *Greedy* and *Linear* start charging immediately. According to the used rate, the first finishes charging after 3 hours whereas the latter finishes just-in-time at 9 hours. All *Convenient* strategies delay the charging. The *Convenient* strategy again finishes just-in-time. As expected, the *Convenient pessimistic* strategies completed charging at 8 hours and the *Convenient optimistic* strategy finishes at 10 hours. All convenient strategies need exactly 3 hours to charge to car. This results state that the model works as expected.

In the following the impact of just-in-time charging on the measures of interest is presented. Section 8.4.1 discusses grid-convenience and Section 8.4.2 compares the integrity probabilities.

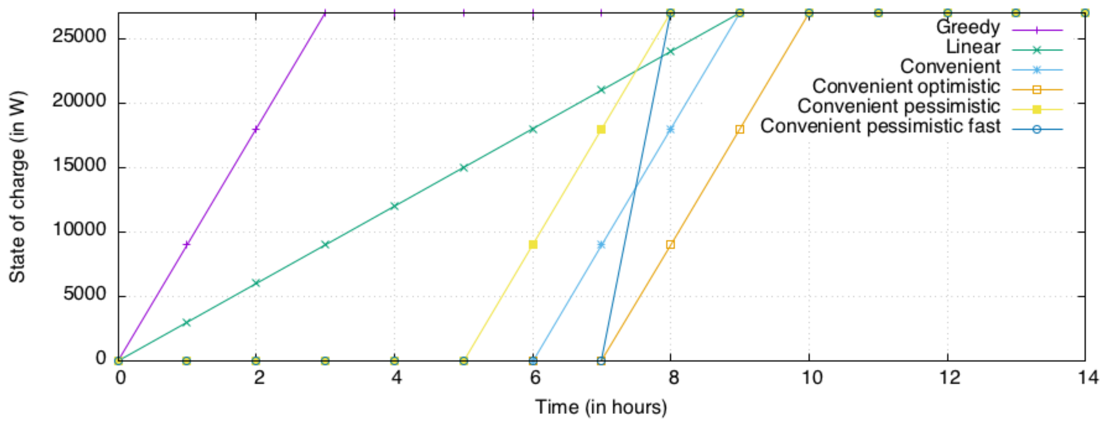


Figure 8.2. The state of charge for the different strategies.

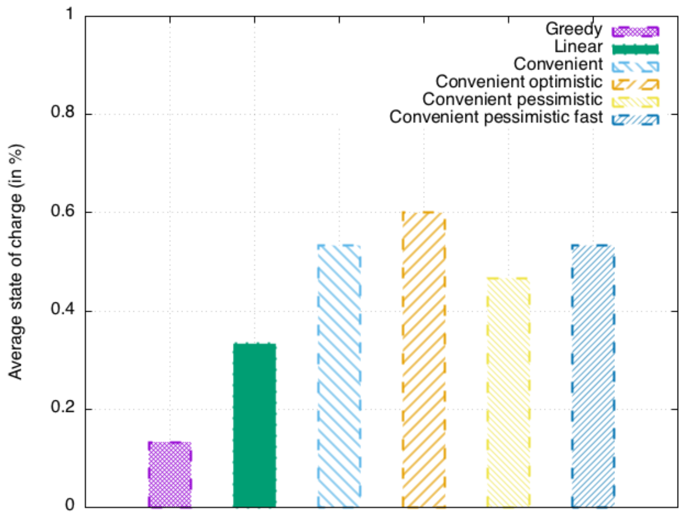


Figure 8.3. Average available battery capacity.

8.4.1 Grid-convenience

As mentioned in Section 8.1 we have to compute the average available battery capacity. A large value is assumed to be grid-convenient. Since the charging strategies finish charging at different points in time, we compute the average available capacity for the interval $[0, 14]$, which corresponds to the maximal charging time. Hence the results will not reach 100% since all stratiegies do not provide any free capacities from at least 9 hours, i.e. when *Convenient optimistic* has finished charging, until 14 hours.

Figure 8.3 presents the average state of charge on the y-axis for all six strategies. As expected *Greedy* provides a very low capacity, since it immediately starts charging and finishes quite early. *Linear* charging doubles the available capacity, but it falls below the *Convenient pessimistic* strategy, which provides an average state of charge of 46%. *Convenient* and *Convenient pessimistic fast* provide an equal average state of charge of 53%. The *Convenient pessimistic fast* strategy performs best, providing in average 60% of its battery capacity.

8.4.2 Integrity

We have defined the property **integrity** that holds in case the electric vehicle is completely charged when the clients returns. The client determines the total time of charging, but its recurrence can slightly differ. As mentioned in Section 8.3 we use several deviations to assess the influence on the used strategies. Figure 8.4 presents the integrity probability on the y-axis. The x-axis represents different standard deviation values in hours.

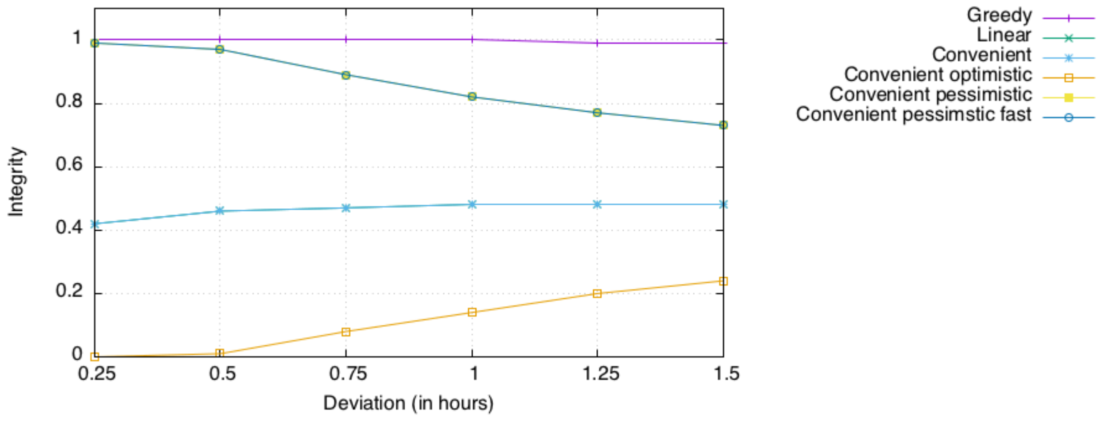


Figure 8.4. Probability of a completely charged EV when the client returns.

As expected, the *Greedy* strategy offers an integrity of around 100% for all deviation values. *Convenient pessimistic* and *Convenient pessimistic fast* perform equal due to the fact that they finish charging at the same time. They provide a good integrity for small deviation values of 0.25 and 0.5 hours. For larger deviation the integrity strongly decreases down to 73% for a deviation of 1.5 hours. Likewise the *Convenient* and *Linear* strategy provide an equal probability for the same reason. However their integrity is much worse. For a deviation of 0.25 hours the integrity is 42%. It slightly increases for higher deviation values, but never exeeds 50%. Since the *Convenient optimistic* strategy assumes a later recurrence of the client, the integrity for a small deviation of 0.25 hours and 0.5 hours is around 0%. It increases up to 24% for a higher deviation, since it becomes more likely that the client recurs later when the car is completely charged.

The pessimistic strategies perform very well for small deviation values, wheareas the integrity of all the other convenient strategies does not satisfy the claim for a charging system. However, we only investigate the probability to have a completely charged strategy. A charging system may also work satisfactorily if the battery is nearly completely charged. Therefore Figure 8.5 presents the probability that the electric vehicle is partly charged. The y-axis represents the probability and the x-axis represents the proportion of charge that is needed to satsify the integrity. We use a fixed deviation of $\sigma = 1$ in this case.

The integrity of *Convenient pessimistic* now overcomes the *Convenient pessimistic fast* a little, since the first starts loading 1 hour earlier and thus is more robust w.r.t. earlier recurring clients. It is very interesting to see, that the *Linear* strategy now performs much better than

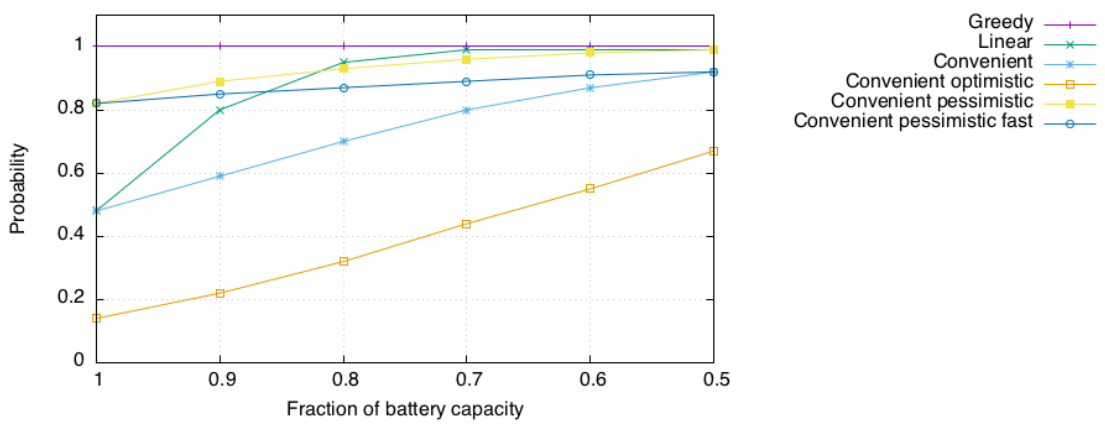


Figure 8.5. Probability of a partly charged EVs battery.

the *Convenient* strategy, and even provides an integrity that is around 100% for a proportion of 80%. This states that it is advisable to start charging early, even with a lower charging rate, to provide high integrity values. Again the results of the *Convenient optimistic* strategy are not satisfactory.

The presented results show that the *Convenient pessimistic* strategy may be the choice to introduce a grid-convenient charging process. It provides the best tradeoff between integrity and grid-convenience w.r.t the investigated strategies. Depending on the reliability of the determined charging time, it provides a very high integrity value. Even in case the client returns earlier, it is likely that the car is highly charged.

8.5 Discussion

The presented Hybrid Petri net model is able to implement several charging strategies regarding electric vehicles. We concentrated on strategies that delay the charging to enable a coordinated use. These strategies were compared to the recently used *Greedy* strategy.

We have seen that whenever the total available charging time is known, it is not needed that the charging has to start immediately. Due to the availability of high quality charging station the charging is that quick, that the car is completely charged very fast. Thus the battery can neither be used as a buffer for load balancing nor is the grid able to introduce a coordinated charging process. This emphasizes the idea of the presented strategies. It is sufficient that the car is charged just-in-time, i.e. as soon as the client returns.

We have investigated the impact of said strategies on grid-convenience and integrity. A delayed charging provides a higher average battery capacity to the grid and thus enables a convenient use. We can state that the *Greedy* strategy focusses only on the consumers needs. It provides a very high integrity probability, but does not enable a grid-convenient use. However, the *Convenient optimistic* strategy is much too grid-convenient and provides a very low integrity probability. The *Convenient pessimistic* strategy turned out to be the best tradeoff between grid-convenience and integrity w.r.t. the investigated strategies.

In contrast to the introduced grid-convenient charging in Chapter 7 the client has to actively support grid-convenience. He has to determine the total charging time and thus consciously is confronted with a possible restriction of his interests. It would be fairly easy to implement a delayed charging, but clients may only support it if they do have advantages. This may be introduced by paying for a delayed charging, e.g. a lower price for charging, the longer the car is available for charging.

9 Discussion and outlook

9.1 Discussion

We investigated the influence of grid-convenience on the clients interest, i.e. self-use, survivability and integrity. Two comparable Hybrid Petri net models were presented in this thesis that are suitable to assess battery charging strategies. One deals with grid-convenient charging within smart homes. The other treats the charging of electric vehicles. It turned out that whenever good predictions exist, grid-convenience may not decline the client's measures of interest.

The presented results regarding the smart home model especially showed that adaptive algorithms must be implemented in order to reach grid-convenience as well as high self-use und survivability values. Smart homes must obtain weather information continuously and adapt the charging strategy based on the current state of charge. Whenever the prediction does not meet the current overflow, the charging strategy has to be changed. Event though this may result in complex algorithms needed to implement such a system.

However, the implementation of algorithms needed to charge an EV grid-conveniently are comparatively easy. The charging only must be delayed in order to enable a coordinated use. Furthermore a bidirectional communication between the grid and the charging station must be possible. The system must rely on the correctness of the determined clients recurrence time. Whenever this prediction is wrong, the system may not work as expected.

Both models treat grid-convenient charging, but try to reach it by different ways. The used strategy within smart homes does not require any interaction with the client. Whenever the system works as expected, i.e. the predictions are accurate, the client even does not notice that the charging works grid-conveniently. The first presented strategy thus uses a *passive* approach. However, the charging of electric vehicles depends on the will of the client to determine the total charging time. Thus it uses an *active* approach to reach grid-convenience.

All in all we want to recommend the implementation of grid-convenience, particularly within smart homes. Algorithms and data analyzation methods are available that are suitable to implement reliable predictions. Likewise highly distributed systems were recently introduced that suffice to provide adequate communication techniques. Furthermore the obtained results show that we may be able to reach grid-convenience without neglecting the clients interests.

9.2 Outlook

Further research may aim at investigating the *Prediction Based* strategy. As stated above, adaptive algorithms are needed. To assess these algorithms by HPnG models, the recent formalism has to be enhanced. For example it may be meaningful to introduce guard arcs between two transitions, e.g. a immediate and continuous one, that enables the immediate transition whenever the continuous' rate exceeds a specified treshold. Thus hereby, for example, events can be fired in case the overflow exceeds a given threshold. This may simplify the implementation of higher adaptive models needed to investigate the *Prediction Based* strategy.

Other research may aim at enhancing the introduced HPnG models. It may be interesting to model smart homes that work independent of the grid. The needed peak production of the local power generation as well as the battery size needed to continuously supply the house with power may be investigated. Furthermore the influence of prioritized demand classes on the survivability can be investigated. This model may be based on the idea, that consumers from low prioritized demand classes can be switched off in case the battery is less charged and the grid failed.

Generally it may be meaningful to implement a discrete event simulator. Using simulation, it is possible to allow more than one general transition. This may lead to more complex models that are suitable to represent real world system more precisely. However, the research can focus on implementing efficient data structures and algorithms that reduce the computational complexity. Furthermore the simulator must include a model checking procedure to be able to investigate system properties.

References

[1] H. Wirth, “Aktuelle Fakten zur Photovoltaik in Deutschland,” *Fraunhofer ISE*, 2015.

[2] T. Trigg, P. Telleen, R. Boyd, F. Cuenot, D. D’Ambrosio, R. Gaghen, J. Gagné, A. Hardcastle, D. Houssin, A. Jones *et al.*, “Global ev outlook: understanding the electric vehicle landscape to 2020,” *Int. Energy Agency*, pp. 1–40, 2013.

[3] H. Ghasemieh, B. R. Haverkort, M. R. Jongerden, and A. Remke, “Energy resilience modelling for smart houses,” in *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2015)*, 2015, pp. 275–286.

[4] Accenture, “How can utilitites survive energy demand disruption?” *Accenture’s Digitally Enabled Grid program*, 2014.

[5] M. Gribaudo and A. Remke, “Hybrid Petri nets with general one-shot transitions for dependability evaluation of fluid critical infrastructures,” in *Proceedings of the 12th IEEE International High Assurance Systems Engineering Symposium (HASE 2010)*, November 2010, pp. 84–93.

[6] H. Ghasemieh, A. Remke, and B. R. Haverkort, “Survivability evaluation of fluid critical infrastructures using Hybrid Petri nets,” in *Proceedings of the IEEE 19th Pacific Rim International Symposium on Dependable Computing (PRDC 2013)*, 2013, pp. 152–161.

[7] J. Struth, M. Leuthold, A. Aretz, M. Bost, S. Gährs, M. Cramer, E. Szczechowicz, B. Hirschl, A. Schnettler, and D. U. Sauer, “Thesen und Hintergründe zum Nutzen von Speichern in netzgekoppelten PV-Anlagen,” 2013.

[8] J. Struth, M. Leuthold, A. Aretz, M. Bost, S. Gährs, M. Cramer, E. Szczechowicz, B. Hirschl, A. Schnettler, D. U. Sauer *et al.*, “PV-Benefit: A critical review of the effect of grid integrated Pv-storage-systems,” in *8th International Renewable Energy Storage Conference and Exibition. Berlin: Institut für ökologische Wirtschaftsforschung*, 2013.

[9] J. Moshövel, K.-P. Kairies, D. Magnor, M. Leuthold, M. Bost, S. Gährs, E. Szczechowicz, M. Cramer, and D. U. Sauer, “Analysis of the maximal possible grid relief from PV-peak-power impacts by using storage systems for increased self-consumption,” *Applied Energy*, vol. 137, pp. 567–575, 2015.

[10] J. Li and M. A. Danzer, “Optimal charge control strategies for stationary photovoltaic battery systems,” *Journal of Power Sources*, vol. 258, pp. 365–373, 2014.

[11] M. Jongerden, J. Hüls, A. Remke, and B. Haverkort, “Assessing the cost of energy independence,” in *Proceedings of the 4th International Energy Conference, ENERGYCON 2016 (accepted for publication).* IEEE, 2016.

[12] M. Sterner, F. Eckert, M. Thema, and F. Bauer, “Der positivebeitrag dezentraler batteriespeicher für eine stabile stromversorgung,” *Forschungsstelle Energienetze und Energiespeicher (FENES) OTH Regensburg, Kurzstudie im Auftrag von BEE e.V. und Hannover Messe, Regensburg / Berlin / Hannover.*, 2015.

[13] R. Liu, L. Dow, and E. Liu, “A survey of pev impacts on electric utilities,” in *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES.* IEEE, 2011, pp. 1–8.

[14] M. F. Shaaban, M. Ismail, E. F. El-Saadany, and W. Zhuang, “Real-time pev charging/-discharging coordination in smart distribution systems,” *Smart Grid, IEEE Transactions on*, vol. 5, no. 4, pp. 1797–1807, 2014.

[15] E. Sortomme, M. M. Hindi, S. J. MacPherson, and S. Venkata, “Coordinated charging of plug-in hybrid electric vehicles to minimize distribution system losses,” *Smart Grid, IEEE Transactions on*, vol. 2, no. 1, pp. 198–205, 2011.

[16] K. Clement, E. Haesen, and J. Driesen, “Coordinated charging of multiple plug-in hybrid electric vehicles in residential distribution grids,” in *Power Systems Conference and Exposition, 2009. PSCE’09. IEEE/PES.* IEEE, 2009, pp. 1–7.

[17] A. Zimmermann, M. Knoke, A. Huck, and G. Hommel, “Towards version 4.0 of TimeNET,” in *Proceedings of 13th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2006.* VDE VERLAG GmbH, 2006.

[18] B. F. Postema, “Fluid Survival Tool: A model checker for Hybrid Petri nets,” Master’s thesis, UT Twente, 2013.

[19] Kreditanstalt für Wiederaufbau (KfW), “KfW-Programm Erneuerbare Energien Speicher (275),” *Merkblatt Erneuerbare Energien.*

[20] “§6 Abs. 2 Nr. 2 EEG 2012.”

[21] Nissan, “Nissan Leaf performance,” accessed 9. März 2016. [Online]. Available: http://www.nissanusa.com/electric-cars/leaf/versions-specs/version.s.html

[22] Tesla motors, “Tesla Model S specifications,” accessed 9. März 2016. [Online]. Available: http://www.teslamotors.com/support/model-s-specifications

[23] CEER, “CEER benchmarking report 5.1 on the continuity of electricity supply, ref. c13-eqs-57-03,” February 2014, accessed 9. März 2016. [Online]. Available: http://www.ceer.eu/

[24] F. N. im VDE, “Versorgungszuverlässigkeit und Spannungsqualität in Deutschland, url=https://www.vde.com/de/fnn/arbeitsgebiete/versorgungsqualitaet/documents/fnn-fakten-versorgungsqualitaet_2013-03-11.pdf.”

[25] A. Avritzer, L. Carnevali, H. Ghasemieh, L. Happe, B. R. Haverkort, A. Koziolek, D. Menasche, A. Remke, S. S. Savestani, and E. Vicario, “Survivability Evaluation of Gas, Water and Electricity Infrastructures,” *Electronic Notes in Theoretical Computer Science 310 (2015) 5–25*, 2015.

[26] R. David and H. Alla, *Discrete, continuous, and hybrid Petri nets.* Springer, 2005, vol. 1.

[27] W. Zuberek, “Timed petri nets definitions, properties, and applications,” *Microelectronics Reliability*, vol. 31, no. 4, pp. 627–644, 1991.

[28] M. K. Molloy, “Performance analysis using stochastic petri nets,” *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 913–917, 1982.

[29] H. Ghasemieh, A. Remke, B. Haverkort, and M. Gribaudo, “Region-based analysis of hybrid petri nets with a single general one-shot transition,” in *Formal Modeling and Analysis of Timed Systems.* Springer, 2012, pp. 139–154.

[30] B. F. Postema, A. Remke, B. R. Haverkort, and H. Ghasemieh, “Fluid survival tool: A model checker for hybrid petri nets,” in *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance.* Springer, 2014, pp. 255–259.

[31] J. Frank, “PENG - Plattformunabhängiger Editor für Netz-Graphen,” Master’s thesis, TU Berlin, 2006.

[32] RapidXml. (2006) Rapidxml. [Online]. Available: http://rapidxml.sourceforge.net

[33] Gnuplot. (1986) Gnuplot. [Online]. Available: http://www.gnuplot.info

[34] N. L. Q&A. (2015) How long does it take to charge the nissan leaf? [Online]. Available: http://www.nissanusa.com/electric-cars/leaf/owner-questions/ev-charge-time/#leafQnADeepLink=true&searchValue=null&filterIndex=0

[35] L. Gilpin. (2014) The state of electric cars: 10 things you should know. [Online]. Available: http://www.techrepublic.com/article/the-state-of-electric-cars-10-things-you-should-know/

[36] P. Connor. (2011) Eight tips to extend battery life of your electric car. [Online]. Available: http://www.plugincars.com/eight-tips-extend-battery-life-your-electric-car-107938.html

[37] National Renewable Energy Laboratory, “PVWatts Calculator,” accessed 9. März 2016. [Online]. Available: http://pvwatts.nrel.gov/index.php

[38] EDSN, “EDSN demand profiles,” accessed 9. März 2016. [Online]. Available: http://www.edsn.nl/verbruiksprofielen/

[39] K. Clement-Nyns, E. Haesen, and J. Driesen, “The impact of charging plug-in hybrid electric vehicles on a residential distribution grid,” *Power Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 371–380, 2010.