UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

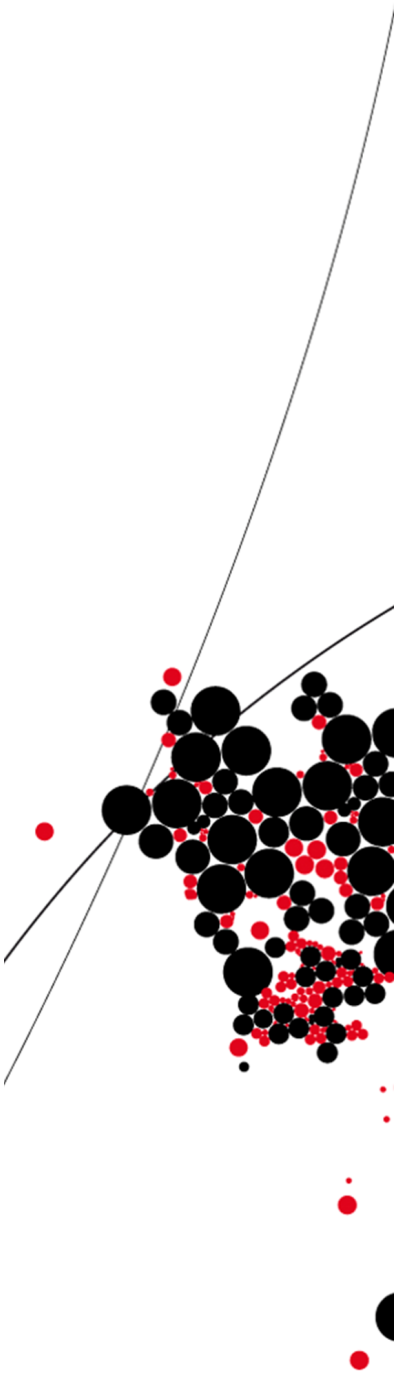
Embedded Infrastructure for Decentralized Energy Management

Thijs Havinga

B.Sc. Thesis

Advanced Technology

July 2019



Assignment committee:
Chairperson: prof.dr. J.L. Hurink
Daily supervisor: dr.ir. G. Hoogsteen
External member: dr. H.K. Hemmes

Discrete Mathematics and
Mathematical Programming
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

Uncontrollable renewable energy sources, such as solar panels and wind turbines, as well as the increasing use of electrical devices leads to a mismatch between energy supply and demand. A solution to this is to regulate the demand side, such that it matches with the variable supply. In order to control the power consumption of devices efficiently, the active control methodology has been developed, for which a dedicated communication infrastructure is needed. The addition of a communication layer to the electricity grid, in order to measure and control the energy usage, forms the smart grid. In this thesis, various approaches for deploying the communication infrastructure necessary for the decentralized control of embedded devices are investigated. The general layout of the communication network in a smart grid is addressed. Relevant communication technologies and protocols are examined using the OSI-model. The requirements for a proper performance of the active control methodology are set up, followed by possible solutions based on these. A Wireless Mesh Network seems a suitable option for connecting smart meters. In order to acquire a robust and flexible system, the paradigm of Software Defined Networking is promising. An emulation is used as a testbed in order to investigate the behavior of the application as if it were deployed in a real communication network. Reproducible experiments using various topologies have been set up. The overall performance of the current implementation complies in great extent to the requirements, but scalability might be the limiting factor. Deliberately inducing network failures resulted in poor performance of the algorithm. An approach for the infrastructure, as well as small modifications on the implementation are given based on literature and experiments.

Acronyms

AMI Advanced Metering Infrastructure.
AP Access Point.
ARP Address Resolution Protocol.

BAN Building Area Network.
BLE Bluetooth Low Energy.

CoAP Constrained Application Protocol.

DAP Data Aggregation Point.
DDoS Distributed Denial-of-Service.

HAN Home Area Network.
HEMS Home Energy Management System.
HWMP Hybrid Wireless Mesh Protocol.

IAN Industrial Area Network.
ICMP Internet Control Message Protocol.
IED Intelligent Electronic Device.
IoT Internet of Things.
IP Internet Protocol.

LLDP Link Layer Discovery Protocol.
LTE Long-Term Evolution.

M2M Machine-to-Machine.
MAC Medium Access Control.
MPLS Multiprotocol Label Switching.

NAN Neighborhood Area Network.

OSI Open Systems Interconnection.

PLC Power-line communication.

QoS Quality of Service.

RTT Round-Trip Time.

SDN Software Defined Networking.
SDSW Software Defined Switch.

TCP Transmission Control Protocol.
TLS Transport Layer Security.

UDP User Datagram Protocol.

WAN Wide Area Network.
WMN Wireless Mesh Network.

ZMTP ZeroMQ Realtime Exchange Protocol.

Contents

Chapter 1

Introduction

In order to reduce greenhouse gas emissions, new sources of electricity such as solar panels and wind turbines are rapidly replacing fossil fuel based power plants. However, the generation of energy with such sources depends on weather conditions. Therefore, the amount of generated electricity cannot be controlled in order to provide exactly in demand. Besides, the amount of electrical powered devices, some of which were formerly powered by other sources, is increasing. Moreover, devices like heat pumps and electric vehicles are often used simultaneously, which burdens the current electricity grid. A solution for these problems is to regulate the demand side. To this end, an algorithm has been developed which optimizes the overall power profile of a set of devices [1]. In this way, the energy consumption can be scheduled efficiently based on the flexibility of a device. The combination of an electrical grid, information flow and the computational elements forms the so-called smart grid. In order to facilitate the bi-directional information flow from the monitoring sensors and controller commands from the computational elements to the actuators, a stable communication network is needed. The focus of this research is on investigating different approaches for deploying the communication infrastructure needed for the decentralized control of embedded devices in smart grids. In order to conduct this analysis, the following questions are formulated:

1. "What is the current approach in terms of methodology and organization of the smart grid architecture?"
2. "Which protocols and services are considered and developed for the use in smart grid communication?"
3. "What are the requirements of a smart grid communication network?"
4. "How can a heterogeneous and large-scale communication structure be tested efficiently?"

These questions are answered first in the following chapters, after which the research topic is addressed. Eventually, a recommendation is given based on the research and experiments conducted.

Chapter 2

Background

2.1 Situation description

There are different possibilities for realizing decentralized energy management. In this section, the specific methodology that is used within this research, its current communication infrastructure and the physical communication topology of the smart grid is described.

2.1.1 Active control methodology

The focus of this research lays on creating an infrastructure for the active control methodology as described in [1]. This is of the two control systems currently implemented in the simulation and demonstration framework DEMKit. Several energy management methodologies exist, which often follow a dynamic pricing approach. Within a time of use pricing program, the price for electricity in certain intervals is published one day ahead, such that costumers can schedule their consumption. The downside of this approach is that there is no direct feedback on what the effect of using these prices was. Due to the fact that customers intend to consume more when the price is low, an expected low demanding interval can turn into the opposite. When controllable devices are available, real time pricing can avoid this problem. With this concept, a price is announced only shortly before the interval takes place. This is also the case for the active control methodology, though here the price is an artificial steering signal. A double-sided auction forms the basis of this methodology. Each device creates a demand function, which specifies the price it is willing to pay for a certain amount of power, taking the preferences of the end-user and the flexibility of the device into account. Some devices can only be in two states: on or off, and thus will create a step function for their demand. Other devices can operate within a large power range and will produce a linear function. Generators, such as solar panels, demand a negative amount of power. The demand functions for these kind of devices and their sum are shown in Figure 2.1.
These local demand functions are first send to a concentrator, which forwards it to an auctioneer. The latter determines a market clearing price based on the sum of the demand functions it receives. This indicates the deviation from the desired behavior of the region the auctioneer operates in. The market clearing price is communicated downwards again, such that the devices know the amount of power they have to consume or produce. The hierarchy of the different stakeholders in this methodology is

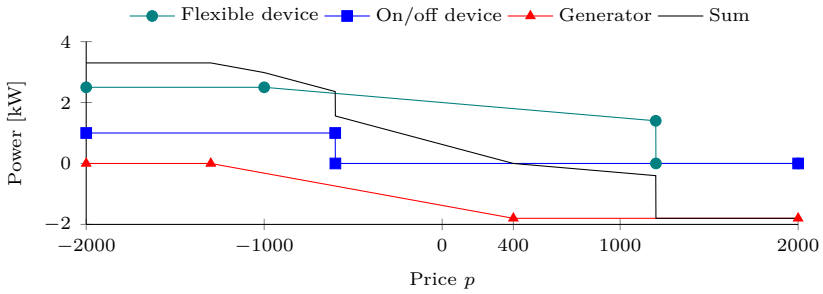


Figure 2.1: Three demand functions for different kind of devices and their sum. Adapted from [1].

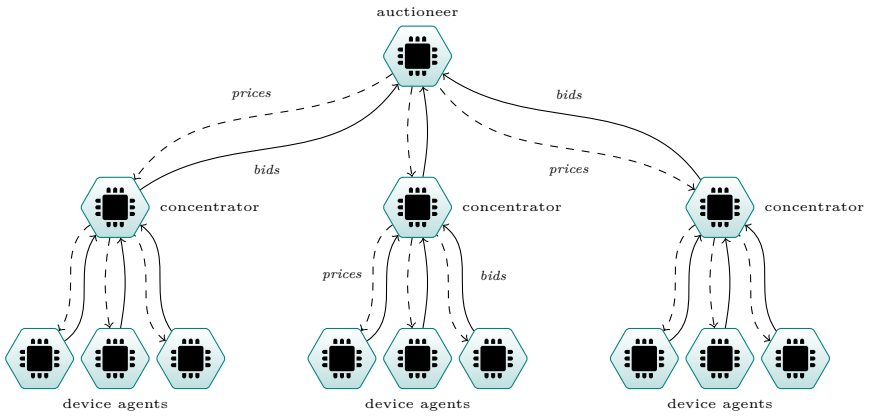


Figure 2.2: Hierarchy of the active control methodology based on a double-sided auction as presented in [1].

shown in Figure 2.2. Demand functions are sent upon request of the auctioneer, for example when significant local changes occurred. The frequency of these bids can become high in case of an islanded microgrid, which requires energy balance at all times.

2.1.2 Topology

A general overview of the communication network architecture of a smart grid is presented in [2]. At the lowest level of the topology of a smart grid, there is the smart home or building with sensors and actuators, connected together in the Home Area Network (HAN). The smart home is assumed to have a Home Energy Management System (HEMS), which collects all sensor data, has a gateway and can control the devices inside the building. A couple of these HEMSs are then connected to a Data Aggregation Point (DAP). The placement of such a DAP in different neighborhoods can influence the routes, transmission rate and energy consumption of the network and is still in ongoing research, e.g. [3] and [4]. Possibly multiple DAPs together form the Neighborhood Area Network (NAN). Each DAP is connected to a master gateway station, which connects to the Wide Area Network (WAN). The two-way communication between the HEMS and higher-level controllers, e.g. connecting HANs to WANs, is often called the Advanced Metering Infrastructure (AMI). A complete overview of the topology within a smart grid is shown in Figure 2.3.
With the double-sided auction approach, the HEMS is the concentrator which collects the demand

functions from individual devices within the HAN. The HEMSs send their collected information to the DAP. Each DAP can perform the function of concentrator as well as auctioneer. This is suitable in the case where the local production suffices, but when the grid is dependent on more remote production of other utilities, the auctioneer should be placed on a higher level. Only one auctioneer per communication network is to be used.
The next two sections will address different networking options on both the HAN level and the levels beyond that.

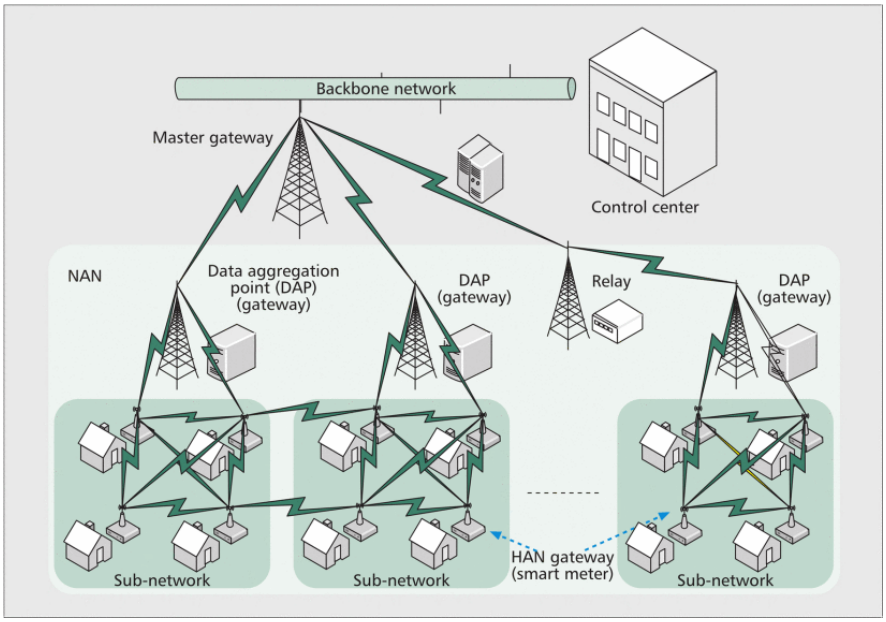


Figure 2.3: Topology of networking in a smart grid, obtained from [2].

2.2 Home Area Networking

Already many different standards for in-home communication exist. The most significant difference between them is the transmission medium used. This can be separated into two categories: wired and wireless mediums. The following section lists these, as pointed out in [5].

Power-line communication (PLC) is considered for wired in-home communication as is does not require to install extra cabling or wireless receiving modules to electrical devices, nor needs the device to be in standby mode in order to be able to receive. Current PLC standard HomePlug 1.0 achieves a data rate of 14 Mbps and it is expected that future standards will reach up to 100 Mbps. PLC works on much higher frequencies than the 50 Hz alternating current power distribution; typically 20-25 MHz is used.
Another technology based on the convenience of existing wiring is ITU-T's G.hn, as described in [6]. This is a general protocol, which works over power lines, coaxial cables, as well as phone lines. It supports IP networking with data rates up to 1 Gbps.
Ethernet over twisted-pair cables, coaxial cables or optical fibers is definitely an option for devices

which are already equipped with it, but installing cables for each device which will connect to the smart grid will be cumbersome.

Wireless solutions include amongst others Wi-Fi, Bluetooth (Low Energy), ZigBee, Z-Wave, and 6LoWPAN. Wi-Fi is already widely used in all kind of (higher-end) devices and allows high data rates. The biggest downside of Wi-Fi for the use in small devices is that it is rather energy-consuming.
Bluetooth Low Energy (BLE) is an adapted version of the Bluetooth standard, which works with a mesh topology and consumes less power compared to conventional Bluetooth and Wi-Fi. However, it can only connect to a maximum of seven nodes at a time.
ZigBee, developed upon the IEEE 802.15.4 standard, shares the low energy consumption and mesh characteristics of BLE. It supports significantly more connections, however, it has only a range of around 10 meters.
Z-Wave is similar to ZigBee, but it can achieve a range of 100 meters in optimal circumstances. A downside of Z-Wave is the lower data rate of around 40 kbps, but for most applications in a HAN, this will suffice.
6LoWPAN is designed to support Internet Protocol (IP) on embedded devices. Therefore, it is optimized for low power consumption and minor processing capabilities. A mesh-based protocol designed for home automation on top of 6LoWPAN is called Thread. It has a range of 20-30 meters, supports up to 250 connections and has a data rate of 250 kbps.
Devices working with one of these protocols can often not seamlessly work together, which is considered as one of the prominent problems of home automation. However, several open source domotica software platforms exist which provide this integration. Examples are the OpenHAB, Domoticz and Home Assistant platforms, as mentioned in [1].

2.3 Networking beyond the HAN

In the following section, possibilities for networking at the levels higher than the HAN are addressed. Unless otherwise stated, a scope no further than the NAN is taken into account. The analysis is done using the Open Systems Interconnection (OSI) model, which is widely used in network systems to partition the different functionalities of a network into layers, see Figure 2.4. This allows to examine all possibilities for communication in the smart grid in a structural way. However, some approaches do not fit directly into this model, therefore they will be described afterwards.

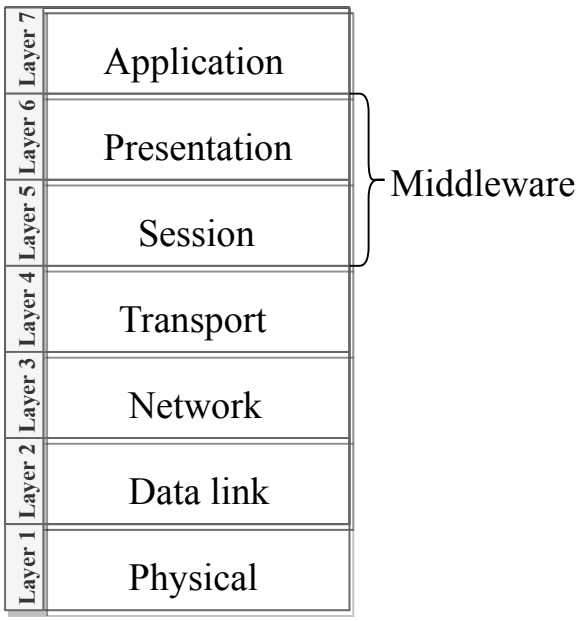


Figure 2.4: Layers of the OSI-model.

2.3.1 Physical layer

Like in the Home Area Network, there are different transmission mediums possible and they

can be divided in wired and wireless options. A summary of what is written in [7] is given below.
Wired solutions used nowadays for the Internet, such as optical fibers and twisted-pair cables seem a suitable choice for covering WAN distances. PLC can be used to facilitate the information flow from HEMSs to DAPs and further as well. However, it is sensitive to errors due to electrical devices connected to it. Also, repeaters are needed in order to carry the signal over a large distance. Next to these problems, security is an issue, as the lines are not shielded and thus can easily be tapped. The latter can to some extend be solved by encrypting the data.
For short-range wireless communication, e.g. up to connecting multiple HEMSs, the physical layer of the IEEE 802.11 standard (Wi-Fi) can be used. Beyond that, longer range technologies such as WiMAX or cellular communication technologies, such as LTE or 5G are needed.

2.3.2 Data link layer

There are different network topologies possible for deployment in a smart grid. As pointed out in [2],[8],[9],[10] and [11], a Wireless Mesh Network (WMN) seems to be a suitable choice for connecting smart meters. In a WMN, each node itself is a router, which forwards packets to its next hop. Multiple routes exist between the nodes. This redundancy makes sure that nodes can still communicate when some of them are offline. A WMN is very scalable due to the self-healing aspect; new routes are created automatically. However, as the routes need to converge, semi-static nodes are required, but this is no issue for a HEMS.
There are different possibilities to acquire a WMN, of which the most are handled in the data link layer of the network. Examples of standards are IEEE 802.11s with the Hybrid Wireless Mesh Protocol (HWMP) and IEEE 802.15.4g, which is designed specifically for Smart Utility Networks.
If a WMN is not possible due to the transmission range or density of smart meters, the data link layer depends on the specific underlying physical layer used. For example, for wired Ethernet links, the data link layer as specified in IEEE 802.3 should be used.
The IEEE 802.11ah standard [12] has been developed especially for the use in the Internet of Things (IoT), such that a large number of stations can be connected, while consuming relatively little energy. The 802.11 standard has undergone modifications on both the physical and Medium Access Control (MAC) layer in order to obtain a range of up to 1 km. This comes at the cost of a lower data rate compared to 802.11s. Instead of creating a mesh, it is designed to connect a large number of devices to one Access Point (AP), using a built-in multi-hop relaying mechanism, but without an extensive network protocol. The 802.11ah standard is considered as a backhaul network for IEEE 802.15.4g, in order to extend its range. Next to that, it can also co-exist with 802.11s.

2.3.3 Network layer

An IP-based network layer is often proposed and assumed in communication for smart grids. Currently it has a low cost of deployment and maintenance and there are various possibilities and implementations to enhance security [13].
There are a couple of different routing protocols considered for smart grids, with an underlying WMN as building block, see [14] and [15]. They differ a.o. in latency, reliability and scalability. As a WMN with only one gateway node might not be sufficient to handle all data, a tree-based protocol is considered to be a more appropriate structure, according to [16]. In this way, the DAP acts as the root

node and different packets (from DAP to meter or from meter to DAP) may take different paths to a destination. In traditional redundant switched networks, the spanning tree protocol is often used. One of the widely considered tree-based protocols for WMNs is HWMP, which is used in IEEE802.11s. There are some adaptations made to HWMP in order to make it more suitable for the use in smart grids. The first one is HWMP with Hybrid Metric, which is a different way of calculating a link's quality [17]. Another adaptation is called Load Aware HWMP [18], which avoids that one link has to forward way more than another by adapting the routes accordingly.
In [19], HWMP is used to mitigate the overhead that is introduced by the Address Resolution Protocol (ARP). ARP makes sure that an IP-address is linked to a physical MAC-address. However, after their modifications the route requests and replies within HWMP can perform the same function.

2.3.4 Transport layer

The two most widely used transport layers are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). As TCP first sets up a connection between the devices and uses acknowledgments to ensure reliable data transfer, it introduces delay and network overhead. However, this is often small compared to the excess introduced by the application. Next to that, a multicast functionality is not provided using TCP.
UDP has much less overhead, but it lacks reliable data transfer. Therefore, as mentioned in [20], a combination between TCP and UDP can be a suitable option for networking in a smart grid. In [21], Split and Aggregated TCP is proposed, which does not ensure end-to-end reliability, but only for each direct connection the delivery is guaranteed.

2.3.5 Middleware

For distributed systems the middleware (sometimes referred to as the session and presentation layers), which lays in between the transport layer and the application layer, is of great importance as well. Most importantly, it assures that the different data representations used by the devices is delivered to the application layer in an understandable way. Next to that, as explained in [22], context awareness, security and possibly network load balancing can be implemented by the middleware as well, if it is not yet obtained by the lower layers.
There are many different middleware platforms available, listed in [23] and [24], which all have different functionalities. To follow the decentralized approach, the middleware has to be Message-Oriented. Devices namely have to perform the calculations locally and communicate the results upwards, upon request of the auctioneer.
Examples of middleware platforms are RabbitMQ, ActiveMQ, ZeroMQ, XMPP, MQTT, Crossbar.io, ICE and Apache Kafka. In [25], a couple of these are compared for the use in smart grids, using different hardware. Among these, ZeroMQ and ICE perform quite well in terms of throughput and latency. There are also a lot of smart grid-specific middleware platforms available. In [26] it is described that those are not yet suitable for real-world deployment, as no effort is done in standardization.

2.3.6 Application layer

Another approach is to handle specific desires for the smart grid at the application layer. These application architectures then rely on underlying protocols such as TCP and IP. Examples of these

standards are OpenADR and IEEE 2030.5, as mentioned in [1], and RIAPS [27].
A connectionless application layer protocol called Constrained Application Protocol (CoAP), which is used above UDP and designed for resource-constrained devices, is considered in [19] for use in smart grids. It is shown that it outperforms UDP and TCP in terms of packet delivery and throughput. Adaptations to CoAP have been made as well, in order to improve its congestion control mechanism.

2.3.7 Multiprotocol Label Switching

A different technique, which works in between the data link layer and the network layer, is to include information about routing inside the packet [28]. Instead of calculating the routes in each hop based on IP addresses, in Multiprotocol Label Switching (MPLS), a label is pushed onto the front of the packet. This label contains enough information for the next router to know where to forward the packet to. At each router, the label is swapped in order to provide the next router with the right information. Neighboring routers have advertised their forwarding criteria on beforehand to each other. Using labels saves considerable time compared to IP-address look-ups. Next to this, packets can be classified based on several characteristics, such that different routes are chosen and Quality of Service (QoS) can be applied.

2.3.8 Software Defined Networking

Software Defined Networking (SDN) is a new paradigm in networking, where a central controller decides on the routes for incoming packets and the underlying switches only forward these packets based on the controller commands. The controller applies the rules it has been given based on the content of a packet.
For several reasons, SDN is a promising technology for the use in smart grids, as pointed out in [29]. It can easily adapt on changes in the network and it is better manageable to the scale and to the different devices that exist in a smart grid. Next to that, it eases the computational burden on embedded network devices, so it enhances efficient use of the hardware resources.
The biggest threat of SDN is the single point of failure of the central controller. Also, the initial request to the controller for the routing of a new stream introduces overhead. However, as shown in [30], substantial improvements on network performance can be achieved with routing by using SDN. Although SDN was originally designed for static wired circumstances, it can be used in WMNs as well, as described in [31]. It is mentioned that the integration can also be done in a hybrid way; traditional routing protocols provide connection between switches and controllers, whereas SDN is used between the switches themselves. Moreover, a hybrid framework can be achieved by using traditional switches and routers in combination with SDN-enabled switches.

Chapter 3

Requirements and approach

In this chapter, the specific requirements for networking in a smart grid are addressed. As the focus lays on applying the active control methodology, these requirements are restricted to the customer domain inside the NAN. Afterwards, the different possibilities that are presented in previous research are considered. Lastly, the performance metrics used to examine the different systems are listed.

3.1 Requirements

The information flows between devices, smart meters and controllers within a smart grid is conceptually different from the Internet used nowadays. Where the latter is focused on supporting multimedia applications over various sessions for different end users, the former has to supply data exchange between multiple devices autonomously. This type of communication without human intervention is also called Machine-to-Machine (M2M) communication. This change in paradigm and its self-ruling nature makes that it comes with several requirements, as listed in [32].
A network deployed for a smart grid must be robust. When a sudden connection fails, all sources still need to be able to reach all destinations and the failing node should be reported. Thus it must be fault-tolerant and self-healing in order to remain stable. As many different devices will be used throughout the grid, the network should be versatile. Furthermore, because there will be no abrupt transition from a traditional to a smart grid, the infrastructure should be scalable. It must be possible to add more and more devices gradually, without degrading the network performance by a large amount. Considering the different environments and surroundings the smart grid will be deployed in, the network needs to be flexible, e.g. adaptations on the network to let it work in a specific environment should not be needed.
Especially in case of an islanded microgrid, the delay of the network should be restricted, since the commands are used for real-time control. Delay leads to a mismatch between the actual power consumption and the outcome of the auction. In order to reduce balancing errors at the auctioneer, which tries to compensate this, the delay should be limited. If the error becomes too large, the grid can even become unstable, which can cause a blackout.
The conceptual difference between conventional communication systems and those in a smart grid generally leads to a demand for more access for the numerous devices, but the data rate is allowed to be lower. In [33], the traffic of the active control methodology, which they call PowerMatcher,

is assessed. Instead of ZeroMQ, the message bus MQTT is used on top of TCP/IP. In their implementation, the publish-subscribe pattern is used between the auctioneer, concentrators and clients as well. From a distribution grid model of 2 million households, they derive a peak load of 54.67 kbps for the auctioneer. A concentrator located in a house with 20 devices would get a load of 1.12 kbps. However, this was modeled using market clearings every 5 minutes, though in DEMKit the default is 1 minute. This number is not fixed and can possibly even be less, because it is market dependent. Next to that, the message size of prices, bids and aggregated bids are estimations and may differ in the implementation of DEMKit. The network traffic of the application scales linearly with the market clearing interval. So, the expected peak load can go up to around 300 kbps using 1 minute market clearings.

The communication infrastructure in a smart grid is not limited to providing data streams for demand response. Monitoring and controlling the electrical transmission and distribution grid are important parts as well. Hence, there are different flows of information, which have distinct demands. Therefore, it is desired to have differentiated QoS that can be applied.

Even though the information flow from within the HAN towards the higher levels is already limited by the decentralized approach, security remains an important aspect. The elements that security should provide are as follows. Data should be integer; it may not be modified by an outsider. Next to that, it should be authenticated, such that it is assured that it comes from the right user. The data should also be confidential, as due to privacy the power consumption of a home should only be granted to trusted utilities. To a certain extend, these requirements can be met by the use of encryption and signing [34]. Another threat for a smart grid network is the lack of availability due to cyber-attacks. For example, a Distributed Denial-of-Service (DDoS) attack can cause that the control of devices cannot be applied anymore, leading to a mismatch of supply and demand. Precautionary measures against these kind of attacks should be taken into account while designing the network.

To summarize, these are characteristics that a smart grid communication network should have or what it is required to be:

- Robust: fault-tolerant and self-healing;
- Versatile: many different devices should be able to connect;
- Flexible: it can be applied in various surroundings;
- Scalable;
- Restricted delay;
- Reasonable data rates;
- Differentiable QoS;
- Secure: integer, authenticated, confidential and available.

3.2 Possibilities

As already mentioned in section 2.3.3, a network for smart grid based on IP is seen as the most viable option. A great and required characteristic of IP is that it can work on various underlying networks.

Namely, considering the variable scale in which the network will be deployed, different physical layers have to be used. It would not be rational to limit to only one specific physical layer, although it might have benefits, because standardizing this will require a lot of modifications to the existing situation and will not be achieved in reality. Among the possibilities, a wireless physical layer and medium access control technology, such as Wi-Fi, Zigbee or Bluetooth, seems a suitable choice for the small scale. However, because its range is limited, on a larger scale Ethernet working on wired mediums or cellular technology is needed. PLC has currently still too many practical issues to be easily set up.
On the scale where short-range wireless technologies as Wi-Fi are available, say up to networking within a NAN, a mesh topology is the most viable option, amongst others because it provides redundancy, as already mentioned in section 2.3.2. A WMN can be achieved in multiple ways, of which IEEE 802.11s' and IEEE 802.15.4's physical and MAC layers both meet the requirements. The former uses HWMP as default routing protocol. As explained in section 2.3.3, adaptations on HWMP are required to make it more suitable for the use in smart grids.
In case a WMN is not suitable, the IEEE 802.11ah standard can be used, or it can operate as a backhaul network. Next to that, the upcoming 5G network also provides opportunities to connect smart meters and aggregators in a wider range.
In order to obtain a robust and flexible network using a high number of diverse devices, the concept of SDN fits well. The reason for this is that the central controller has an overall view, which can optimize the routing more efficiently. A controller can also apply QoS better using the information it has obtained. Next to that, it can more efficiently re-configure the network when needed. In [29], it is mentioned that all aspects of MPLS can be integrated with SDN and that the latter offers more flexibility. For these reasons, it is interesting to investigate the performance of SDN compared to various traditional networking techniques for the use in decentralized energy management. Currently, the opportunities of SDN in a WMN and the performance of hybrid networks are not yet researched extensively.
Because IP is designed to offer a best effort service, it is by itself still unreliable. A dedicated transport layer is therefore needed, but that comes at the cost of overhead. However, reliability can also be achieved on higher layers. Therefore, the transport layers, as listed in Section 2.3.4, in combination with middleware and possibly even application layers can be investigated in order to fulfill the requirements.
In order to comply to the security requirements, several options are available and one is not limited to a single approach, as they can also work alongside each other. It can be implemented in one of the layers, whilst SDN offers both possibilities and threats on security.
To summarize, the rest of this research focuses on investigating traditional routing protocols and routing based on SDN for decentralized energy management. Next to that, it will be examined in what way the other requirements can be achieved. This will be done by methods used in the transport layer, middleware, application layer or a combination of them.

3.3 Performance metrics

In order to examine the characteristics of a proposed system, both quantitatively and qualitatively, as to check if it fulfills the requirements that were set up, relevant metrics need to be defined.
Measuring the throughput seems evident, however it makes sense to make a distinction between

the throughput for the application, e.g. the actual 'goodput', and the packets sent needed for the communication to work.
As the final goal is to provide a reliable structure for the application, it is good to measure the amount of requests for bids that are actually delivered and answered. Not only is it interesting to test if packets arrive, the time it takes to achieve this is as well. The Round-Trip Time (RTT) is a measure for the time it takes for packet and acknowledgment delivery between two endpoints. Next to that, the irregularity in RTT is something to take into account; this variation in delay is also called jitter. The application should namely be able to catch up with the expected maximum deviation of the RTT.
Since one of the requirements is that the network should be fail-safe, in case of a link failure, the system should be able to recover. The time or amount of packet losses it takes to recover is a good metric for the self-healing performance of the network. Also, if a link comes back online, the network should adapt to achieve the original structure again. The overhead that is introduced by configuration changes and its influence on the performance can also be investigated by measuring the request delivery ratio, RTT and jitter.
It is difficult to test the security of a network, because there are no real metrics for it. Therefore, different options can only be compared in terms of the ease of deployment of precautions such as encryption and signing and via an evaluation of the risks that come with the system. Further work needs to be done on this topic, because it falls outside the scope of this research.
Thus, the performance metrics that will be taken into account are as follows:

- Throughput and goodput;
- Amount of delivered requests;
- RTT and jitter;
- Link recovery time;
- Failure overhead.

Chapter 4

Research methodology

This chapter elaborates on the methodology to investigate the possibilities for achieving a communication structure that complies to the requirements as described in the previous section. At first, the approach to set up and investigate the performance of several systems is presented. Then a theoretical explanation of SDN is given. Afterwards, the application used to test the active control methodology is described.

4.1 Network emulation

As it is impractical to set up an environment which satisfies the conditions and correctly reproduces the surroundings of a NAN for a smart grid, a virtual framework suits this research better. In addition, it is less costly and more convenient to scale the structure up to a realistic scenario and to examine different configurations. Next to that, it gives more opportunities for benchmarking the system, because it is reproducible. However, since virtualization does not have the same characteristics as the real-life implementation, the features of the physical hardware have to be simulated. Networking simulations are based on the theoretical performance of devices and protocols. It models the behavior of components, which will thus inherently not completely represent reality. However, considerable work is done in validating simulators, such that they are accepted to be used in research, see e.g. [35]. Moreover, the test should be representable for the actual situation in the sense that the active control methodology can really function within the framework. Hence, the available system as used in the simulation should also be ready to be implemented on a real machine. This is the principle of emulation; multiple instances of end-hosts, which can perform the same functions as their physical real-life counterparts, are executed on an emulation server. Therefore, real packets of the application are send over the underlying layers of a virtual network. These packets are thus sent in real-time and therefore, the emulation is constrained to use lower link speeds when the CPU resources are limited. One of the most widely used emulators is Mininet [36]. It makes use of lightweight virtualization, in order to have multiple hosts working in the same way as the real machine. For each host, a shell can be used, which shares the file system of the emulation server, but has its own network namespace. The switches and links are implemented in software, such that e.g. link failures can easily be simulated. Topologies and other characteristics of the network can be programmed using Python scripts. Both traditional networking elements and SDN components can be used. For traditional networking, the

switches are nodes that use the bridge implemented in the Linux kernel, which forward packets based on Ethernet addresses. To connect different networks with each other, a Linux router is used, which performs IP-forwarding. How SDN is implemented in Mininet, as well as its general concept (from [28]), is explained in Section 4.3.
Mininet-WiFi [37] is an extension of Mininet, especially for the use of wireless links. It uses the *mac80211_hwsim*, which is a Linux kernel module consisting of wireless device drivers, to simulate Wi-Fi radios. Mesh networks are also supported, for which it uses *open80211s*, an open source implementation of the IEEE 802.11s standard.

4.2 Packet analysis

Since the implementation of the active control methodology relies on underlying protocols, it is important to investigate the network on lower levels. This can be done by looking at the raw packets that are being sent over the interfaces. A packet analyzer, or 'sniffer', is a tool that captures network traffic and dissects each packet. In this way, the relevant information that a packet carries can be inspected. Fields that can be exposed in this way include the protocol that is used, the source and destination IP- or MAC-address, the packet length, the actual data and protocol specific information. Also, because timestamps are used, statistics such as delay and throughput can be derived from a capture. The packet analyzer used in this research is Wireshark [38].

4.3 SDN approach

In SDN, there is an evident separation between the control plane and the data plane. Normally, both planes are implemented in a router, where the control plane decides on the routes and the data plane forwards the incoming frames. On the other hand, using SDN, a controller carries out the functionality of the control plane and the data plane is realized by simple switches. The most common protocol that is used between the controller and the switches is OpenFlow, which uses Transport Layer Security (TLS) that runs over TCP. Each OpenFlow switch holds a flow table with three entries: match fields, counters and instructions. Like in MPLS, packets can be categorized based on its fields, such as input port, Ethernet, IP and TCP/UDP addresses and IP type of service. The amount of packets that match a flow is registered in the counter. The instructions are the actions that a switch has to perform when a packet matches a certain flow. These include forwarding the packet to a certain interface, to a port or to the controller, enqueuing or dropping the packet or modifying a field.
In a larger network, usually multiple controllers are used. This increases reliability and reduces the computational load per controller. In [39], a theoretical model for the number of switches per controller is given, considering a total processing time that is acceptable for the application. Next to that, based on the previous result, they give the optimal number of controllers if the amount of smart meters in a certain area is known. They provide an overview of multiple connected NANs when using SDN, which is shown in Figure 4.1. Every smart meter or Intelligent Electronic Device (IED) within the HAN, Building Area Network (BAN) and Industrial Area Network (IAN) is linked to a Software Defined Switch (SDSW), which are connected to a local controller, that in turn connects to higher-lever controllers.
Mininet uses Open vSwitch, which models OpenFlow switches in software. As a controller, the

Openflow reference controller can be used, but one can also connect to a remote controller, running somewhere in the network. There are various programmable SDN controllers available, such as POX, Ryu, OpenDaylight and ONOS.
Apart from programming them manually, the POX controller has a number of built-in components. The *openflow.discovery* component lets switches use the Link Layer Discovery Protocol (LLDP) protocol in order to detect the links in the network. A component that ensures that every packet has to match a flow exactly is called *forwarding.l2_learning*. If the packet does not match exactly, a new flow is created for it.

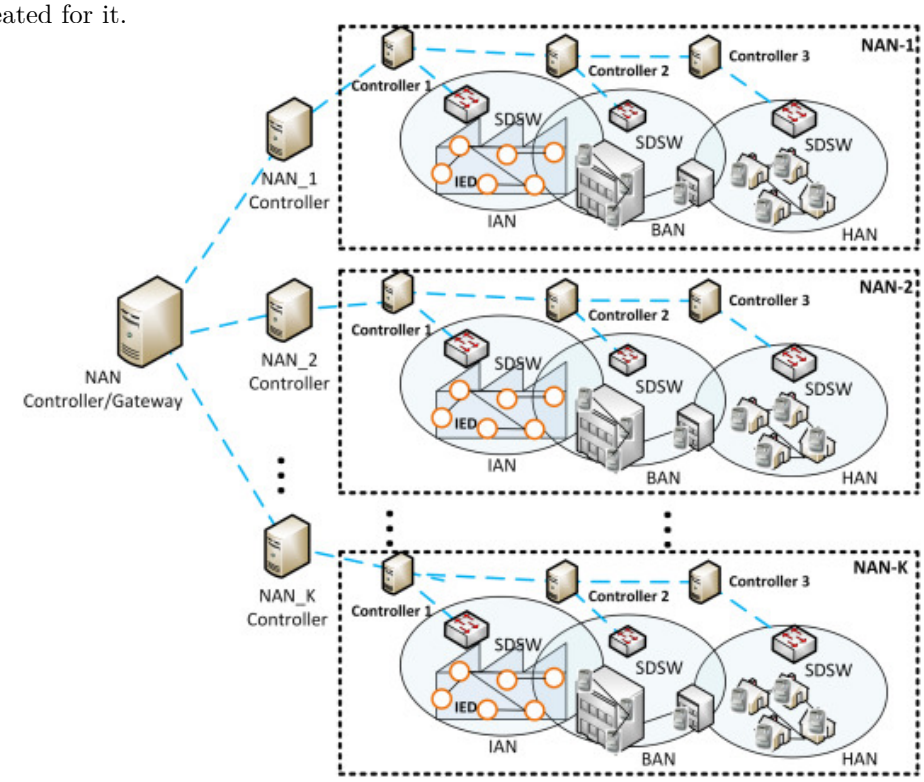


Figure 4.1: Placement of SDN controllers and switches in NANs.

4.4 Application

The implementation that is currently used in the DEMKit platform assumes an IP-based network. On top of TCP, the distributed messaging library ZeroMQ [40] is used. The functionalities of ZeroMQ that raw TCP lacks are amongst others asynchronous I/O handling, intelligent message queuing and network error handling. Rather than a one-to-one connection, it has different patterns that can be used to deliver the messages.
DEMKit uses a message bus which subscribes to every kind of message from every client and can publish messages itself. The clients, which are both devices and auctioneers, publish and subscribe only to the bus. This pair works asynchronously: the client is constantly waiting for something to receive, while the message bus only sends a packet if it needs to. This publish-subscribe pattern does not know if a client is still there or not. Therefore, the clients send a periodic signal, a so-called heartbeat, in order to notify the message bus (and possibly other clients) that the client is still alive.

Apart from connecting devices directly to the message bus of the auctioneer, there is an option to use concentrators. They publish and subscribe to both the master bus, to which the auctioneer is connected, and a slave bus, to which clients of the concentrator are connected. The concentrator aggregates the bids of his clients and forwards this to the master bus, such that the auctioneer can select a price. Usually, concentrators collect data from multiple buildings, but a smart meter can also be seen as a concentrator, because it aggregates the bids of the domestic assets. In this way, multiple levels of concentrators can exist, which communicate the aggregated bid one level upwards. See Figure 4.2 for an overview of the Publish-Subscribe pattern of ZeroMQ between the various components in DEMKit.

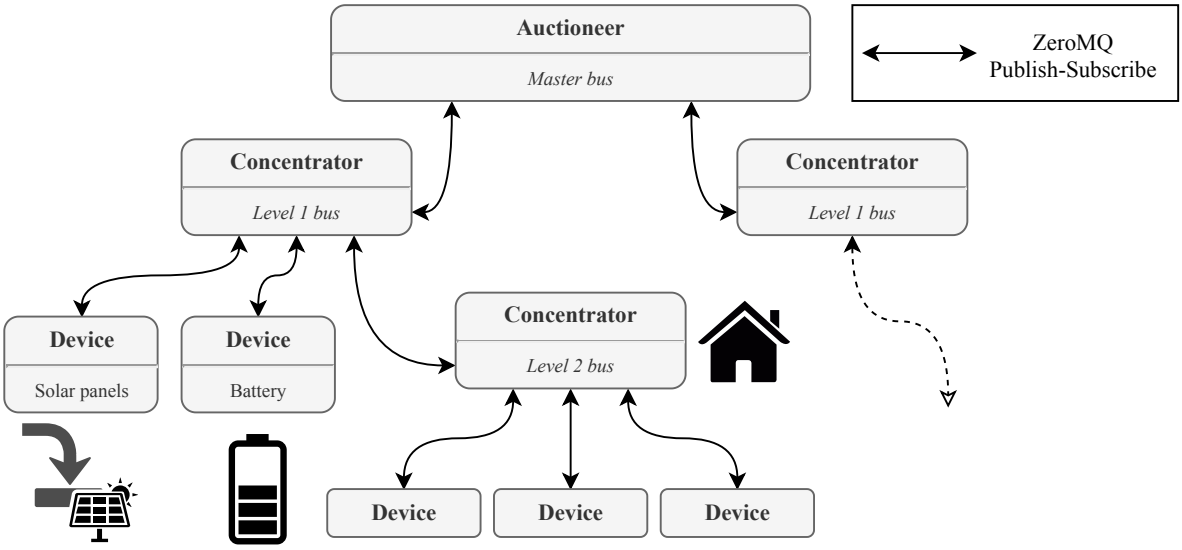


Figure 4.2: Example of the ZeroMQ publish-subscribe pattern between the components in DEMKit.

Chapter 5

Experimental Analysis and Results

5.1 Aims

Multiple experiments are conducted by emulating the implementation of DEMKit on various hosts in different situations, in order to test the requirements listed in Section 3.1. These can be quantified using the performance metrics specified in Section 3.3. At first, the specific performance of the transport layer and middleware is investigated. Afterwards, the effect of using concentrators in a large network is examined. Then one of the most important requirements, robustness, is evaluated while comparing SDN with traditional networking. To conclude, the influence of a failing network on the performance of the algorithm is investigated.

5.2 Topologies

Different set-ups were created in order to examine the various aspects described above. The topologies used during the tests are presented hereafter. An example of how such a topology can be realized in Mininet using a Python script is shown in the Appendix.

5.2.1 Traditional networking

To investigate the working of the application using traditional networking, a topology of three switches (*s1*, *s2*, *s3*), each connected to three devices (*h1*-*h9*) was created in Mininet, see Figure 5.1. These devices, which are Mininet hosts, were connected to the switch via Ethernet links with a bandwidth of 100 Mbps. The switches are modeled as Linux Bridges, connected with a preconfigured default route to a Linux router (*r0*). The auctioneer (which performed market clearings every 10s) and the message bus ran on *h10*.

5.2.2 Wired mesh

In order to examine the performance of a mesh configuration, the topology shown in Figure 5.2 was used. Still the same Ethernet links are used, however, there are some redundant links between the switches. In order to prevent packets from looping endlessly through the network, the spanning tree protocol is used. A switch running this protocol detects if the same packet runs multiple times over

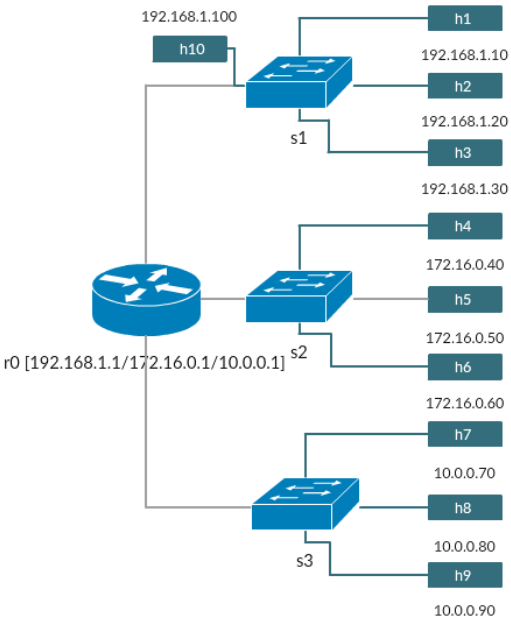


Figure 5.1: Topology used for traditional networking in Mininet.

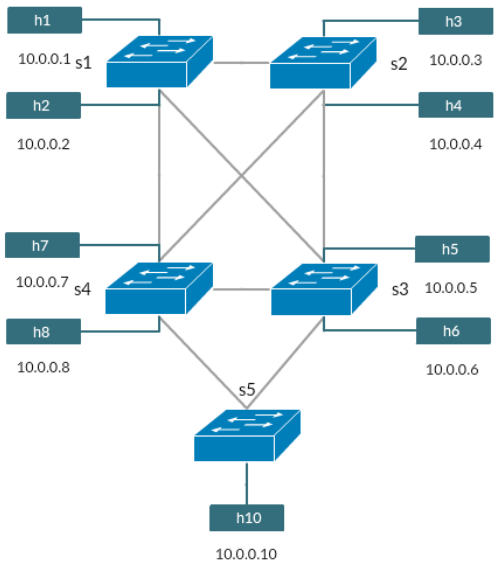


Figure 5.2: Mesh topology used in Mininet.

the same link. Based on this, it blocks some of its ports in order to break the loop. Either the version as it is implemented in the Open vSwitch software for the SDN-enabled case, or the implementation in the Linux kernel is used.

Next to this, a larger wired topology, consisting of 40 switches was created. This structure is based on the topology in Figure 2.3. Groups of four switches were connect in a mesh topology, which then were connected via two links to the next group, see Figure 5.3. To ease the load on the CPU, the application of the device only ran on *h40* and the bus and auctioneer ran on *h1*. Again, both Open vSwitch and Linux Bridges were used.

5.3 Experiments

5.3.1 Transport layer and middleware

Description: First of all, emulation experiments are executed in order to examine the working and performance of the application on top of the transport layer TCP and the middleware, ZeroMQ in this case.

Experiment 1: It is known that both TCP and ZeroMQ initially set up a connection using a couple of packets. TCP uses its handshake and ZeroMQ initializes the connection conform the specifications of the ZeroMQ Realtime Exchange Protocol (ZMTP) [41]. So, it is expected that some overhead is introduced in order to obtain reliability. In order to address this quantitatively, the relative goodput can be expressed by the amount of packets from the application itself as a percentage of the total amount of packets sent.

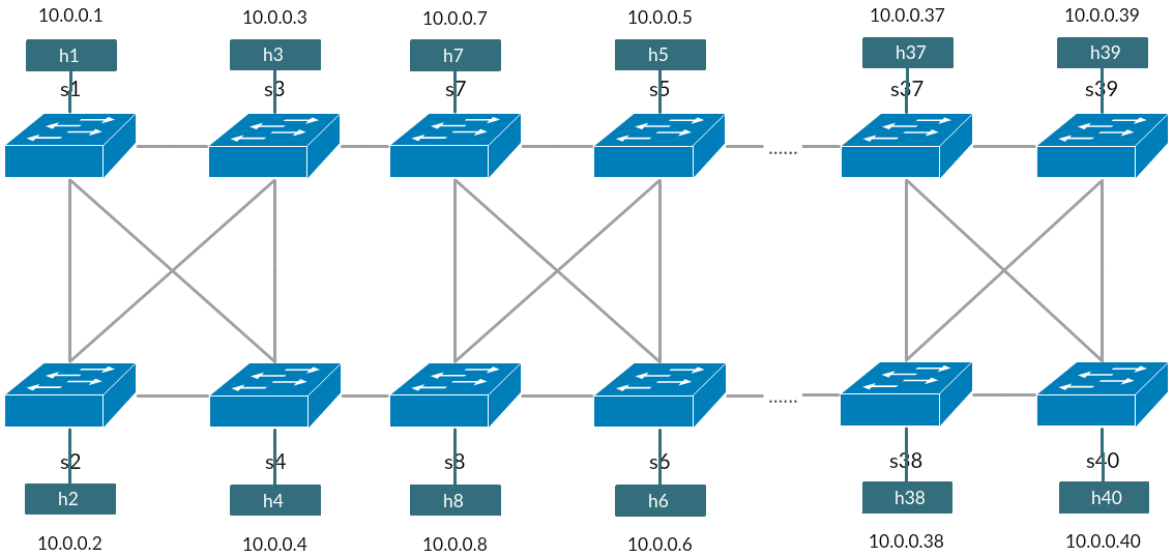


Figure 5.3: Large scale wired mesh topology created in Mininet (12 of the 40 switches are shown).

In this test, the start-up phase of the devices and auctioneer was analyzed during an emulation of 140 seconds using the topology of Figure 5.1. After around 10 seconds, all devices were connected to the auctioneer. By examining the packets, the overhead of the connection set up of ZeroMQ can be analyzed. The connection between clients and the bus is expected to start with a TCP-handshake. When the TCP connection is set up, the ZeroMQ greeting is performed, in order to negotiate on the protocol version. Each ZeroMQ command is encapsulated in the data section of an individual TCP frame. Afterwards, the ZeroMQ handshake is executed. This consists of a security mechanism with additional information about the pattern (Publish-Subscribe in this case) that is going to be used. In the current implementation, the simplest security mechanism is used, which does not provide authentication, nor confidentiality.

Results: See Figure 5.4 for the result. The packets captured are categorized into TCP, ZeroMQ and application. The first kind are pure TCP packets, without a data field, so the packets used for the handshake. The second kind are the packets containing ZMTP commands and the third are the application messages such as bids, prices and the heartbeat. Packets which are received and sent to the devices and the auctioneer are split up. For all devices, the amount of packets for each category was nearly the same. In total, 54% of the packets are for TCP, 35% are ZeroMQ packets and the actual goodput is only 11%.

Discussion: Thus, quite some overhead is introduced by both TCP and ZeroMQ. Due to the initial connection set-up, the relative overhead is only slightly more than on the long run, because the TCP handshake and ZeroMQ greeting is performed for each message. Although the extra packets burden the network, it makes sure that the communication is reliable. Compared to the load on IP-based networks nowadays, the overhead is still reasonably low.

Experiment 2: According to the ZeroMQ guide, if a message cannot be sent, it will be buffered into memory. This is done up to a certain amount of messages, which will be send on a later time, when that is possible. It is interesting to assess how this feature works out for the double-sided auction by

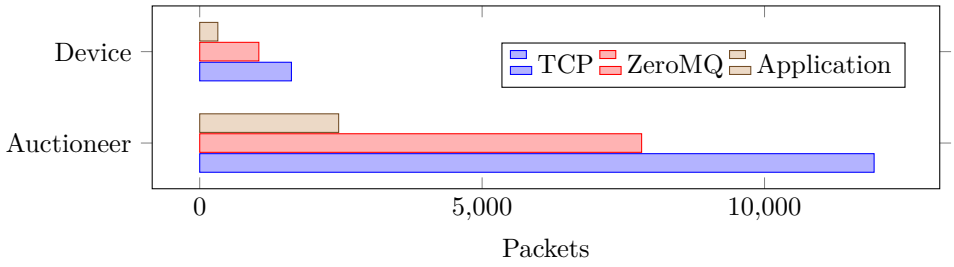


Figure 5.4: Amount of packets of different categories sent by and to the devices and auctioneer.

looking at the number of requests and heartbeats sent out.

In this experiment, the link between *h6* and *s2* in the topology of Figure 5.1 was set down at 30s. After 50s, the link was set up again. During this period, Wireshark captured the packets sent over the virtual interfaces. Especially the TCP packets on the ports that the auctioneer uses are of interest now.

Results: See Figure 5.5 for the amount of bytes sent to and received on *h6* during this period. Measurements are done every 100 ms. It can be seen that there are a lot of bytes sent after the connection is recovered again. This is partly because TCP connections are still alive and the non-acknowledged packets are retransmitted, and partly because ZeroMQ has stored some messages and resent it.

All the messages that are sent at that point are heartbeat messages. This is because during the disconnected phase, the auctioneer still sent out the requests for bids (the small bumps after 30s). It then receives via Internet Control Message Protocol (ICMP) messages from the router that the destination could not be reached. As a reaction to that, TCP tries to retransmit the message about three seconds later. This process repeats for a while, whilst new requests for bids do not get into the queue. On the other hand, the device notices that it cannot send the heartbeat directly, because already the first link it has to use is down. It then keeps on generating heartbeat messages, which it buffers until the connection recovers.

Moreover, it can be seen in the figure that after the link failure the pattern is somewhat different, but this is due to the fact that the request for a bid and heartbeat are not synchronized anymore.

Discussion: Thus, it has been shown that buffering only works if the link error occurs in a single hop. However, it is not useful to buffer and resend the heartbeat, as the response of the device is only relevant if it is received directly. Though, resending them might burden the network, therefore it would make sense to only buffer requests for bids and market prices. Yet, this has to be done for the current market clearing only, because the device cannot react to prices in earlier intervals anymore.

Experiment 3: As explained in Section 3.1, normally the delay requirement is not too strict, though especially in case of an islanded microgrid, it should be within reasonable limits. Since an IP-based network is assumed, other traffic might be present on the links as well. Therefore, it is logical to test the delay of the application also under loaded circumstances. The delay will be expressed in the RTT of a packet. The difference between the average RTT and the RTT during load is the jitter in this case. Wireshark calculates the RTT of a packet by measuring the difference between the time it was sent and the time when the corresponding acknowledgment was received.

In the following experiment, the link between *h6* and the auctioneer in the traditional set-up was artificially loaded using the tool *iPerf* [42]. For 20 seconds, the tool tries to transmit an unlimited

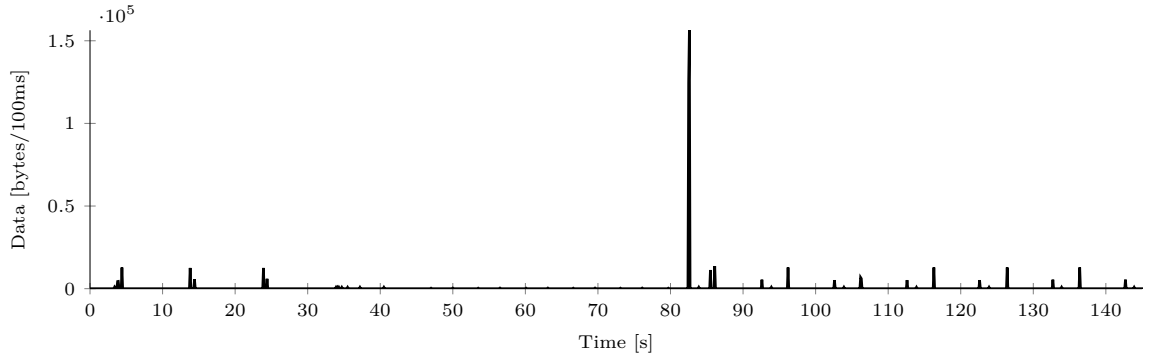


Figure 5.5: Amount of bytes sent to and received on *h6* over time before, during and after a link failure.

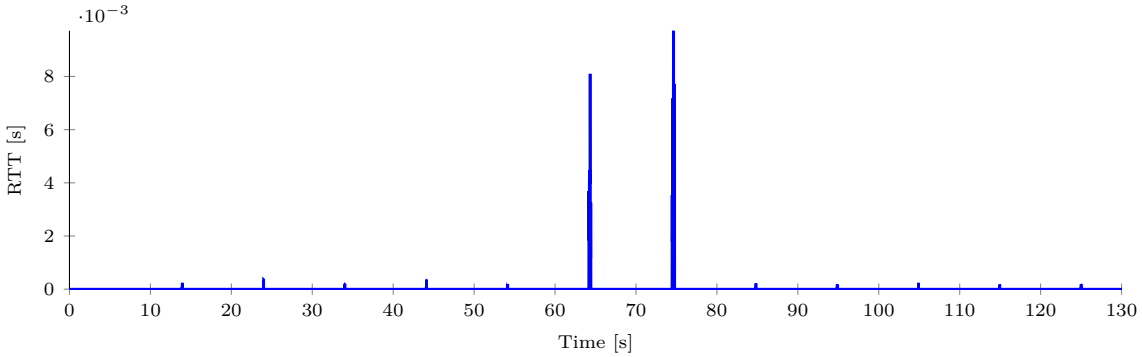


Figure 5.6: RTT of packets sent to and received on *h6* over time before, during and after link overloading.

amount of TCP packets, in order to simulate a very busy link.

Results: See Figure 5.6 for the RTT of packets sent by the auctioneer and the device in a test where the loading started at 62s. The physical distance between hosts introduces additional propagation delay, which is not taken into account in this emulation. However, in the NAN scale, this will be only in terms of microseconds per link. As can be seen in the figure, the RTT is normally less than 0.5 ms, but it induces jitter of around 9 ms during the loading. In the course of multiple tests, the delay during load remained in between 7 ms and 14 ms.

Discussion: Compared to the average delay on a free link, the delay is increased substantially. However, for applications that are not time-critical, like the doubled-sided auction, the packets are still being sent with reasonable delay, even on a very busy channel. Though, in this emulation the loading was introduced on a single link, but the communication infrastructure for a smart grid can comprise large cities with thousands of households. Therefore, messages might cross thousands of links before it ends up at a concentrator or auctioneer and then the accumulated delay becomes significant. Consequently, devices might miss out on bid requests or the clearing interval becomes larger than it ought to be.

Table 5.1: Amount of data and load sent to the auctioneers (and concentrators).

Without concentrators	Data [bytes]	Load [kbps]
Auctioneer	554300	40.31

With concentrators	Data [bytes]	Load [kbps]
Auctioneer	175067	12.73
Concentrator 1	135692	9.87
Concentrator 2	145424	10.58
Concentrator 3	146320	10.64
Total	602503	43.82

5.3.2 Use of concentrators

Description: Concentrators, which are placed in between the devices and the auctioneer, aggregate the bids they receive in order to form one combined bid, which they send to the auctioneer. If bids have overlapping parts, the data can be compressed. Therefore, the aggregated bid generally comprises less data than the sum of all individual bids. Taking the overhead determined in Section 5.3.1 into account, the shrinkage of data is even more. Therefore, it is expected that the use of concentrators as an intermediate point eases the load on the auctioneer and let messages of individual clients travel a shorter path through the network. The load can be expressed by the amount of data that arrives per second at the auctioneer or concentrator. This can then be compared to the estimated traffic from [33], as mentioned in Section 3.1.

Experiment 1: The first test comprised of running the double-sided auction in the large mesh topology of Figure 5.3, with and without concentrators. The market clearing interval was set to 10s. The auctioneer was connected to switch *s15* and in total three concentrators were connected to *s5*, *s25* and *s35* in order to distribute the links between devices equally. Ten devices were connected per concentrator or auctioneer. The amount of data sent to the auctioneer and to the concentrators was inspected. For both cases, this way done during the same period of 110s after all devices were connected.
Result: See Table 5.1 for the amount of data sent to the auctioneer and concentrators in both cases. Also, the resulting load per agent is shown.
Discussion: It should be noted that extra TCP packets that were sent due to e.g. retransmissions or out-of-order packets were not taken into account.
The amount of data that is sent to the auctioneer is more than three times less in the case where concentrators are used. However, as expected, this data is now sent to the concentrators. In total, more data is exchanged using concentrators, which is caused by the extra aggregated bids coming from the concentrators. So, using concentrators indeed spreads the load across the different collectors and is a way to use the network capacity more efficiently. Though, it uses slightly more data in total. In order to compare the estimated load on a concentrator to the 1.12 kpbs mentioned in Section 3.1, this number has to be scaled to 10 devices and a market clearing interval of 10 seconds. This gives a load of 16.8 kbps. In this experiment, the concentrators experience a load of 10.36 kbps on average. The peak load at the auctioneer cannot be compared directly, because in the model that was used in [33], the auctioneer has to aggregate data of a larger and different structure.

Experiment 2: Next to the advantage of load-balancing mentioned above, using multiple concen-

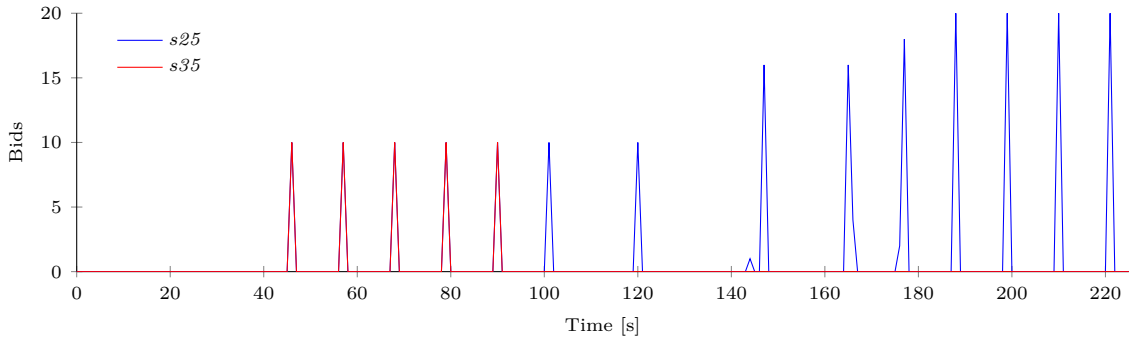


Figure 5.7: Amount of bid functions sent to the concentrators at *s25* and *s35*.

trators can solve the problem of having a single point-of-failure. If a link between a device and concentrator or the concentrator itself fails, the device should be able to connect to another concentrator. This would also comply to the decentralized approach. Therefore, it is useful to investigate how this can be realized in the current implementation.

In order to let devices connect to another concentrator when the former fails, the application has been modified in several ways. Each device now gets a list of IP-addresses and names of the available concentrators. After a certain amount of unanswered heartbeats, it concludes that the connection is lost and it will stop the subscription to that concentrator. It then looks for the next concentrator in the list and asks for a connection with it.

The adapted version was deployed on the network of Figure 5.3. Again, three concentrators and one auctioneer were used, which all connected to ten devices. After a while, the link between the concentrator and *s35* was set down.

Result: After the connection to the concentrator at *s35* was lost, the devices tried to connect to the concentrator connected to *s25*. Figure 5.7 shows the amount of bid functions sent to the concentrator at *s25* and at *s35*. It can be seen that the devices slowly change their connection to the concentrator at *s25*. About 90s after the connection loss all devices reconnected successfully. While the devices were reconnecting, the interval at which the concentrator at *s25* collected the bids became two to three times as big, but this resolved to the original interval after all devices were connected.

Discussion: Using multiple concentrators adds additional dependability for the double-sided auction to perform well. Currently, the various concentrators have to be assigned to the devices statically. A possible improvement would be that a device discovers the concentrators in his neighborhood and then automatically connects to the one which has the lowest cost. Load-Aware HWMP, mentioned in Section 2.3.3, uses a metric that can determine this cost, thereby taking the load at the concentrators into account.

5.3.3 Robustness using SDN and traditional networking

Description: In order to test the robustness of the application running on different kind of networks, introducing a link failure seems a suitable method. Those may be caused by failing or faltering equipment. In case of wireless technology, causes for link failures are channel interference and signal refraction or scattering due to physical obstacles. If redundant links exist, the network is expected to reconfigure and reconnect within a reasonable time. The link recovery time is therefore used as a

metric for the robustness of the system.
It is expected that a traditional network will react differently compared to a SDN-enabled one. During the disconnected phase, the switches need to detect the lack of information frames from the spanning tree protocol from the disconnected link. Using the default parameters, this can take up to 50 seconds. Afterwards, a topology change notification is sent out, which reduces the aging time of entries in the table. This induces the spanning tree to be renewed. Contrarily, the *openflow.discovery* component of the SDN controller first detects via LLDP messages that the link cannot be used anymore. Then the flow table within SDSWs first has to be modified in consultation with the controller before the messages will end up at the right place. It is interesting to see the performance of both implementations on a different network scale.

Experiment: A link failure was induced in the wired mesh configuration of Figure 5.2 between *s2* and *s4* after 63s. In the large mesh network, one of the links from the switch to which *h40* was connected was set down. To enable SDN, SDSWs were used and the POX controller ran the *openflow_discovery* and *forwarding_l2.learning* components on the emulation server (see Section 4.3). For the legacy network, Linux Bridges were applied. Both switches ran the spanning tree protocol, which sends information frames at the default interval of two seconds. The spanning tree can be analyzed by executing the command `ovs-ofctl show` on each switch. This shows a table containing which interfaces are used and which are blocked.
Results: See Figure 5.8 for the amount of packets sent to and received on the devices for both the SDN case (**a**) and using Linux Bridges (**b**) in the small topology. It can be seen that the connection between the auctioneer and *h3* and *h4* was lost for about 20s more for the SDN case compared to the traditional configuration. The same test was performed three times and this resulted in an average connection loss of 49s for device 3 and 54s for device 4 with the SDN architecture. Using the Linux Bridges, the downtime was on average 32s and 36s, respectively.
The spanning tree showed that initially, *s2* sent packets directly towards *s4*, but after the link failure, it utilized the link via *s1*. In the SDN-enabled case, it takes more time before the topology change notification is sent out. However, after this happened, in about 1 second, packets are exchanged again. For the legacy switches, this happens after about 7 seconds.
Regarding the large network, a different result was retrieved. In the SDN-enabled network, the heartbeat discovered that the device was not connected for a while, but the application did not miss any market clearings. That indicates a disconnection time of less than 10 seconds. However, for the large legacy network, the average time that the device was disconnected was 67s. This is caused due to topology change notifications sent out late, as well as the time it took to set up a new tree.
Discussion: Depending on the size of the network, matching flows is quicker than setting up a new tree. In an even more complex network, where a link failure might influence the routes of multiple smart meters, the difference can be quite significant. Moreover, the controller that is used in this experiment is rather straightforward, as it requires that all flows match exactly. This can be optimized by matching only those entries that the messages have in common, such that related packets are joined in one flow.

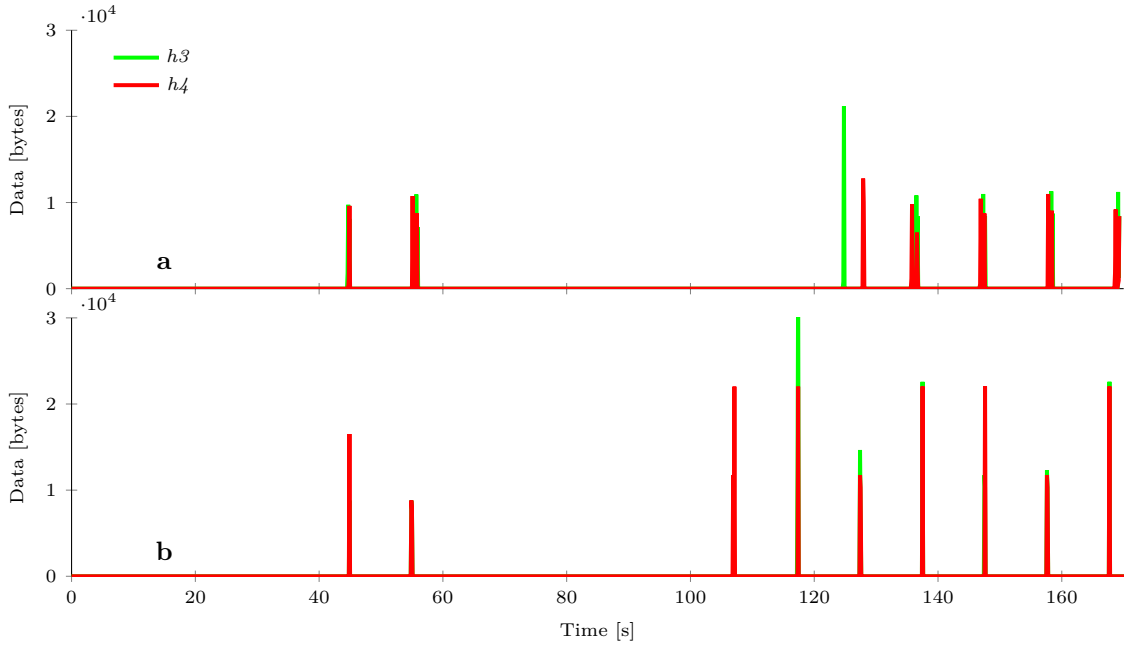


Figure 5.8: Amount of bytes sent to and received on *h3* and *h4* in the topology of Figure 5.2 before and after a link failure between *s2* and *s4* for both the SDN case (**a**) and using Linux Bridges (**b**).

5.3.4 Influence of network failures

Description: A network failure, caused by e.g. failing equipment or a cyberattack, will influence the algorithm. The auctioneer does not receive all information in this case and will therefore base its clearing price on only a subset of the demand. In order to investigate the impact on the power consumption, a house with load devices, solar panels and a battery, which were all connected to an auctioneer, are emulated. During the experiment, the communication between the clients and auctioneer will be disconnected.

Experiment: Measurement data of a real house is used for the desired power consumption of the devices and the production of energy from solar panels. The battery that is modeled is a Tesla Powerwall 2, with a maximum capacity of 13.5 kWh and a maximum (dis)charge rate of 5 kW [43]. The auctioneer tries to clear the market at a total power consumption of 0 W.
The devices form one demand function together, which consists of a small uncontrollable section and a larger flexible part. They only accept their total desired power consumption for very low prices. The solar panels are willing to offer all power they produced, except for a small range of low prices, in which curtailment will be applied. The battery wants to charge when the price is low, discharge when it is high and takes no action for prices in between. It depends on the state of charge of the battery at which rate it demands to charge or discharge. It starts at a state of charge of 0 Wh. Clients that are disconnected from the auctioneer will consume or produce the amount of power that they would do without control. The auctioneer clears the market once every 5 minutes, although in the emulation this is set to 5 seconds, in order to speed up the process.
The situation is emulated for one whole day. After interval 133, the connection between the auctioneer

and the load devices was lost. The devices were back online at interval 161. Then at interval 175, the solar panels are disconnected and from interval 203, they were connected again.

Result: Figure 5.9**a** shows the emulation without disconnections. In Figure 5.9**b**, the disconnection and reconnection of the load devices and the solar panels, respectively, is shown. Only the interesting part from interval 100 until 270 is shown. Outside these intervals, the power consumption is roughly zero.

In **b**, it can be seen that as soon as the load devices are disconnected, the battery charges following the offered power by the solar panels. Therefore, the netto power consumption follows the consumption by the load devices. Hence, this power must come from other sources. The battery also does not compensate for the peaks around interval 150.

During the time that the solar panels are disconnected, according to the auctioneer, the battery has to provide all the needed energy. Since the capacity of the battery is limited, the auctioneer tries to restrict the power usage of the devices in this situation.

It can be seen that in both cases, the total power consumption drops back to zero soon after the connection is established again. However, the state of charge of the battery will be different, because it did not discharge while the load devices were offline and it could not charge while the solar panels were disconnected.

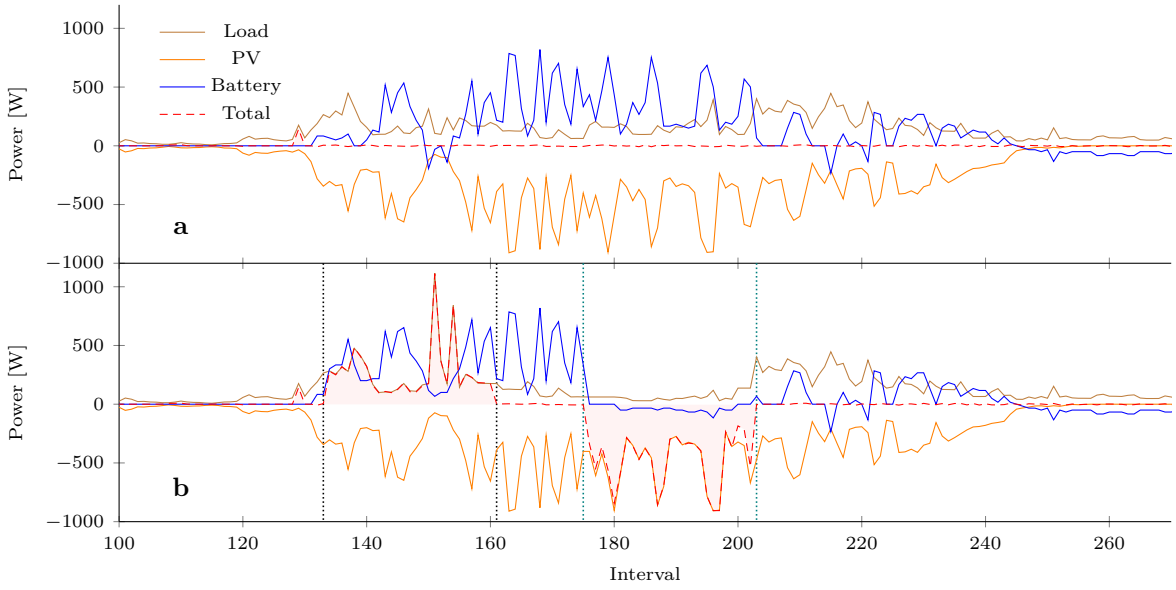


Figure 5.9: **a)** Power consumption of the load devices, PV panels, battery and their sum with normal control. **b)** Power consumption of the clients and their sum during a network error at the load from interval 133-161 and at the PV panels from 175-203.

Discussion: During the disconnected phases, the total power consumption deviates from the desired 0 W by the amount that the disconnected clients consume or produce. This misbehavior might even be less favorable than no control and should be avoided. The auctioneer can find out when a device is suddenly disconnected, but it is difficult to say if this is on purpose or not. In order to implement error handling, a client and auctioneer should come to an agreement in case the client wants to close the connection. Then there are several options to undertake when a real communicational error

occurs. If electrical measurement equipment is installed in the relevant building, this can be used by the HEMS to calculate a local market clearing, as described in [1]. Nonetheless, this requires the HAN to work, such that the clients can still be controlled. If this is not the case, a client can also consume or produce a certain amount based on the latest prices it received or on a pattern from earlier similar periods. This can work decently for a short period, but cannot be used on the long run, because the system is too dynamic.

Chapter 6

Recommendations

In this chapter, a perspective and recommendations for the choices to be made regarding communication in smart grids, especially for the deployment of the active control methodology, are presented. Next to that, discussion points on the conducted research and several suggestions for future work are given.

6.1 Network technologies and layout

Inside buildings, most likely wireless technologies, such as Z-Wave and ZigBee, are to be deployed in order to let devices communicate with the HEMS. As mentioned in Section 2.3.2, several reasons exist why connecting smart meters in the NAN should be done using a Wireless Mesh Network. Standards such as IEEE 802.11s and IEEE 802.15.4g can be used for this. In order to extent the transmission range for less dense neighborhoods or in order to connect multiple NANs, other technologies such as Long-Term Evolution (LTE) or its successor 5G will be needed. A hierarchal tree structure is more suited on this scale. IEEE 802.11ah might be appropriate to deliver data up to the WAN, but needs further research before it can be realized.

6.2 Emulation

An emulator has shown to be a valuable testbed for experiments using virtual hosts with real networking capabilities. Therefore not only the algorithm itself is put to test, but the underlying communicational layers are as well. In this research, experiments are conducted using wired topologies only. However, as a WMN seems suitable for the NAN scale, such a structure needs to be investigated as well. As described in Section 2.3.3, various modifications on HWMP are proposed, indicating potential improvements for the usage in smart grids. Mininet-WiFi can be utilized to experiment with these novelties in the IEEE 802.11s standard before deploying them in a real-world scenario.
Experimenting with large-scale topologies was restricted to about 40 hosts, since the laptop on which the emulations are performed has limited resources. Next to that, due to the same reason, emulating overloading was only reasonable on a single link. Hence, the functioning of the system on the scale where it is to be deployed could not be tested. Using dedicated hardware to perform such experiments

is thus recommended.

6.3 Transport layer and middleware

Middleware is needed for a stable connection within the smart grid, because it controls the information flow between the distributed heterogeneous devices. As mentioned in Section 2.3.5, ZeroMQ performs well compared to other middleware and thus seems a suitable option.
From the result of Experiment 1 in Section 5.3.1 follows that both TCP and ZeroMQ use quite some additional packets in order to deliver the messages. At this point, there is a trade-off between overhead and reliability. If a less reliable transport layer, like UDP, is used, the reliability should be implemented on the application level. This can be done by resending a request for a bid if after a certain timeout still nothing is received. It depends on the frequency of such errors if this is acceptable, because retransmitting in this way will be less efficient than implementing this on lower levels. Another application layer solution that can be examined is CoAP. As mentioned in Section 2.3.6, it is designed to achieve reliability on top of UDP with limited overhead.
Currently, the heartbeat is implemented on application level. TCP also has a *keepalive* option which sends a frame of minimum size only after the connection is idle for a certain adaptable time. Therefore, it is expected that the peak as shown in Figure 5.5 will not occur, since this is caused due to overhead and working of ZeroMQ. However, due to time constraints this is not implemented and thus not investigated.
The security provided by ZeroMQ is not yet used. By employing a more comprehensive handshake, a connection is authenticated and the message data encrypted. This is one of the security measures necessary to be implemented when the application is to be realized for public use.

6.4 Active control methodology

From Section 5.3.2 follows that under normal circumstances the active control methodology performs as expected and the network traffic comes close to estimated values of earlier research. Concentrators can be used to balance the load of network traffic among different agents and are necessary on a large scale. As mentioned in Section 2.1.2, the placement of DAPs — in the context of the active control methodology these will be the concentrators — can be optimized. In Section 5.3.2, it has been shown that devices can reconnect to a different concentrator in case of a network failure. This gives an opportunity to let smart meters connect to the concentrator that suits them best, for example by balancing the load among them.
As shown in Section 5.3.4, not adapting the algorithm during communicational errors and thus working with incomplete data will result in suboptimal control. A closing handshake between clients and auctioneer is needed in order to distinguish between a deliberate disconnection and a network error. Several actions can be undertaken when an error is detected. For example, in case the HAN is still running, the HEMS can produce a local clearing price using meter data from additional measurement hardware. Research can be done into other options for error handling.

6.5 Software Defined Networking

In literature, the potential of using SDN within the smart grid often comes up. Currently, research and experiments are going on in order to apply modifications to controllers and the OpenFlow protocol in order to deploy it in a WMN. From the experiment in Section 5.3.3, it followed that on a large scale, a SDN-based network might have advantages compared to traditional networking. However, it can be seen in Figure 5.6 that the extra delay due to e.g. overloading on a single link is not disturbing for the active control methodology. Again, on a large scale SDN can have more effect, because the delay adds up and can become significant. As described in [29], other applications in the smart grid, such as the automation in electrical substations can be very time-critical and are therefore even more suited for SDN. If a SDN infrastructure is to be realized for those kind of applications, the active control methodology may benefit from it as well. In this research, the potential of SDN was not entirely reached, because only a simple controller that was available was used. In order to make SDN successful, research can be done in order to create controllers that provide QoS to various smart grid applications.

The use of a central controller seems contradictory to the idea of decentralized energy management, but using multiple controllers a hierarchal structure, such as the one given in Figure 4.1, can be obtained. The amount of controllers and switches can be optimized using the analytical model given in [39]. However, the costs of deployment and maintenance also have to be taken into account. A hierarchal SDN approach is also legitimate, because it is similar to the topology of the double-sided auction. When choosing the amount and placement of controllers, the distribution of concentrators need to be taken into account as well.

Using multiple controllers also reduces the threat of a single point-of-failure. On the other hand, SDN can contribute to fulfill the security requirements, because it makes implementing policies better scalable. Using flows, a flexible firewall can be implemented in order to manage inter-domain communication. Since security was not the focus of this research, no further effort was done on this topic, but it remains an important aspect.

Chapter 7

Conclusion

The communicational organization of a smart grid generally consists of separate networking areas which are interconnected. Devices in the HAN exchange information and are controlled by the HEMS. Wireless technologies especially designed for small radios are often used on this scale. Multiple HEMSs, possibly connected to DAPs, form the NAN. Researches suggest using a WMN for this area. The layer above, the so-called WAN, consists of multiple NANs, control centers and electric utilities which are connected via a backbone network, realized by either cellular or wired technologies.
One of the considered control mechanisms for decentralized energy management is the active control methodology, which is based on a double-sided auction. It fits well in this concept, as all kind of clients can be connected to concentrators, forming multiple layers in a hierarchal way.
Among the requirements of a smart grid communication network, versatility and flexibility come up because of the heterogeneous infrastructure. The current communication module within the active control methodology, which is based on IP and uses middleware for distributed devices, already follows the right approach in order to obtain these requirements. Additionally, deploying a WMN in combination with a dynamic routing protocol provides the required robustness.
In order to experiment with communicational elements running smart grid applications, it is useful to arrange a realistic structure in an inexpensive and quick way. A network emulator suits this purpose, since it generates multiple virtual hosts and networking components as if it were separate physical devices.
Experiments performed with the active control methodology show proper overall performance on a rather large scale. However, inducing network failures caused poor behavior of the algorithm. Therefore, it is important to keep track of the availability of devices and to handle accordingly in case of errors.
Furthermore, when implementing the communication infrastructure for a smart grid in a real neighborhood and beyond, one might face scalability issues. These concerns can be practical, such as managing the QoS for various applications and administering the numerous devices. To this end, the concept of SDN offers several effective means. Its flexible nature allows for easier configuration of switches in order to provide QoS and execute policies. Moreover, the performance is expected to reduce on a large scale, because then delay, overhead and load imbalance becomes more significant. Dynamic reconfiguration of routes using SDN flows might mitigate the first issue. Reconsidering specific implementation details could reduce the problem of overhead. Balancing of network traffic load

can be achieved by concentrators, of which the placement should be optimized. Besides, the clients could be associated with the right concentrator in an intelligent way.
Part of the security requirements, such as confidentiality and integrity, can be achieved by the middleware. However, providing a secure platform, which is also insusceptible for deliberate attacks, remains an open issue. Decentralized energy management simply cannot be realized if the privacy of end-users cannot be guaranteed, nor if the system is not reliable.
To summarize, in this paper, an outline for the infrastructure needed for decentralized control in a smart grid is given. Emulation is proposed as a matter to investigate the working of a specific application on top of a realistic communication network. In general, the performance of the active control methodology is sufficient, although careful choices have to be made in order to make it scalable. Different methods are available to acquire a stable, robust and secure communication network. Nevertheless, proper handling in case of network failures should be thought of.

Appendix

Emulation example code

```
#!/usr/bin/python
from mininet.net import Mininet
from mininet.nodelib import LinuxBridge
from mininet.node import RemoteController, OVSSwitch, Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

def exampleNet():

    # Declare Mininet environment with external controller (can also be set to None)
    net = Mininet( controller=RemoteController )

    info( '*** Adding hosts...\n' )
    auctioneer = net.addHost( 'auctioneer', ip='10.0.0.100')
    device = net.addHost( 'device', ip='10.0.0.1')

    # Specify IP-address and port for the external controller
    controller = net.addController('controller', controller=RemoteController, ip='127.0.0.1', port=6633)

    info( '*** Adding switches...\n' )
    # A traditional Linux Bridge with the Spanning Tree Protocol (STP)
    s1 = net.addSwitch( 's1', cls=LinuxBridge, stp='yes' )
    s2 = net.addSwitch( 's2', cls=OVSSwitch) # An OpenFlow enabled switch

    info( '*** Creating links\n' )
    # Create an Ethernet link, specifying the bandwidth in Mbps
    net.addLink( auctioneer, s1, bw = 100 )
    net.addLink( device, s2, bw = 100 )
    net.addLink( s1, s2, bw = 100 )
```

```
30      info( '*** Starting network\n')
31      net.start()
32      controller.start()
33
34      info('*** Configure switches\n')
35      s1.cmd('ifconfig s1 10.0.1.1') # Specify its IP-address
36      s2.cmd('ifconfig s1 10.0.1.2')
37      # Enabling STP on the Open vSwitch
38      s2.cmd('ovs-vsctl set bridge s2 stp-enable=true')
39
40      # Running bus and auctioneer on the specific host in the background,
            using its IP and name
41      auctioneer.cmd('nohup python3 bus_master.py -i '+auctioneer.IP()+' -n
            bus_'+auctioneer.name+' &')
42      auctioneer.cmd('nohup python3 auctioneer.py -i '+auctioneer.IP()+' &')
43      info("*** Auctioneer started\n")
44
45      device.cmd('nohup python3 device.py -i '+device.IP()+' -s '+auctioneer.
            IP()+' -n '+device.name+' -b bus_'+auctioneer.name+' &')
46      info("*** Device started\n")
47
48      # Link failures etc. can be implemented in code or in the Command Line
            Interface (CLI)
49      s1.cmdPrint('ip link set s1-eth1 down')
50
51      info( '*** Running CLI\n' )
52      CLI( net )
53
54      info( '*** Stopping network' )
55      net.stop()
56
57  if __name__ == '__main__':
58      setLogLevel( 'info' )
59      exampleNet()
```

Bibliography

[1] G. Hoogsteen, "A cyber-physical systems perspective on decentralized energy management," Ph.D. dissertation, University of Twente, Netherlands, 12 2017, CTIT Ph.D. thesis series no. 17-449.

[2] W. Meng, R. Ma, and H. Chen, "Smart grid neighborhood area networks: a survey," *IEEE Network*, vol. 28, no. 1, pp. 24–32, January 2014.

[3] Y. Y. e. a. G. Wang, Y. Zhao, "Data aggregation point placement problem in neighborhood area networks of smart grid," *Mobile Networks and Applications*, vol. 23, pp. 696 – 708, 2018.

[4] F. Aalamifar and L. Lampe, "Cost-efficient QoS-aware data acquisition point placement for advanced metering infrastructure," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6260–6274, Dec 2018.

[5] J. O. Morales, N. A. T. Lopez, J. Parado, and J. R. Pasaoa, "A comparative study of Thread against ZigBee, Z-Wave, Bluetooth, and Wi-Fi as a home-automation networking protocol." 2016. [Online]. Available: http://rgdoi.net/10.13140/RG.2.2.36693.22249

[6] A. Kailas, V. Cecchi, and A. Mukherjee, "Chapter 2 - a survey of contemporary technologies for smart home energy management," in *Handbook of Green Information and Communication Systems*, M. S. Obaidat, A. Anpalagan, and I. Woungang, Eds. Academic Press, 2013, pp. 35 – 56. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124158443000024

[7] A. Mahmood, N. Javaid, and S. Razzaq, "A review of wireless communications for smart grid," *Renewable and Sustainable Energy Reviews*, vol. 41, pp. 248 – 260, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364032114007126

[8] M. R. Abid, A. Khallaayoun, H. Harroud, R. Lghoul, M. Boulmalf, and D. Benhaddou, "A wireless mesh architecture for the advanced metering infrastructure in residential smart grids," in *2013 IEEE Green Technologies Conference (GreenTech)*, April 2013, pp. 338–344.

[9] S. Xu, Y. Qian, and R. Qingyang Hu, "Reliable and resilient access network design for advanced metering infrastructures in smart grid," *IET Smart Grid*, vol. 1, no. 1, pp. 24–30, 2018.

[10] A. Zaballos, A. Vallejo, and J. M. Selga, "Heterogeneous communication architecture for the smart grid," *IEEE Network*, vol. 25, no. 5, pp. 30–37, Sep. 2011.

[11] Z. Bojkovic and B. Bakmaz, "Smart grid communications architecture: a survey and challenges," 04 2012, pp. 83–89.

[12] M. A. Fayyaz, "Networking capabilities of IEEE 802.11s and IEEE 802.11ah systems," 2016.

[13] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A survey on smart grid communication infrastructures: Motivations, requirements and challenges," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 5–20, First 2013.

[14] D. F. Ramírez and S. Céspedes, "Routing in neighborhood area networks: A survey in the context of ami communications," *Journal of Network and Computer Applications*, vol. 55, pp. 68 – 80, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804515000843

[15] N. Saputro, K. Akkaya, and S. Uludag, "A survey of routing protocols for smart grid communications," *Computer Networks*, vol. 56, p. 2742–2771, 07 2012.

[16] H. Gharavi and B. Hu, "Multigate communication network for smart grid," *Proceedings of the IEEE*, vol. 99, no. 6, pp. 1028–1045, June 2011.

[17] Y. Zong, Z. Zheng, and M. Huo, "Improving the reliability of HWMP for smart grid neighborhood area networks," in *2016 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE)*, Oct 2016, pp. 24–30.

[18] A. Robertsingh, D. Devaraj, and R. Narmathabanu, "Development and analysis of wireless mesh networks with load-balancing for AMI in smart grid," in *2015 International Conference on Computing and Network Communications (CoCoNet)*, Dec 2015, pp. 106–111.

[19] S. Tonyali and K. Akkaya, "A scalable protocol stack for IEEE 802.11s-based advanced metering infrastructure networks," in *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2018, pp. 1–6.

[20] N. Kayastha, D. Niyato, E. Hossain, and Z. Han, "Smart grid sensor data collection, communication, and networking: a tutorial," *Wireless Communications and Mobile Computing*, vol. 14, no. 11, pp. 1055–1087, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcm.2258

[21] T. Khalifa, K. Naik, M. Alsabaan, A. Nayak, and N. Goel, "Transport protocol for smart grid infrastructure," in *2010 Second International Conference on Ubiquitous and Future Networks (ICUFN)*, June 2010, pp. 320–325.

[22] J. Rodríguez-Molina, *The Role of Middleware in Distributed Energy Systems Integrated in the Smart Grid*, 07 2016.

[23] M. Albano, L. L. Ferreira, L. M. Pinho, and A. R. Alkhawaja, "Message-oriented middleware for smart grids," *Computer Standards and Interfaces*, vol. 38, pp. 133 – 143, 2015.

[24] L. Magnoni, "Modern messaging for distributed sytems," *Journal of Physics: Conference Series*, vol. 608, p. 012038, may 2015.

[25] B. Petersen, H. Bindner, B. Poulsen, and S. You, "Smart grid communication infrastructure comparison-for distributed control of distributed energy resources using internet of things devices," *International Journal of Electrical and Electronic Engineering and Telecommunications*, vol. 7, no. 1, pp. 7–14, 2018.

[26] J. Rodríguez-Molina and D. M. Kammen, "Middleware architectures for the smart grid: A survey on the state-of-the-art, taxonomy and main open issues," *IEEE Communications Surveys and Tutorials*, vol. 20, pp. 2992–3033, 2018.

[27] G. Karsai, A. Dubey, S. Lukic, and A. Srivastava, "Resilient information architecture platform for smart grid," https://riaps.isis.vanderbilt.edu/, 2019, [Online; accessed 02-05-2019].

[28] G. Bernstein, "A Modern Course on Broadband Networking," [Accessed 5-06-2019]. [Online]. Available: https://www.grotto-networking.com/BBNetworking.html

[29] N. Dorsch, F. Kurtz, H. Georg, C. Hägerling, and C. Wietfeld, "Software-defined networking for smart grid communications: Applications, challenges and advantages," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov 2014, pp. 422–427.

[30] A. Montazerolghaem, M. H. Yaghmaee Moghaddam, and A. Leon-Garcia, "OpenAMI: Software-defined AMI load balancing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 206–218, Feb 2018.

[31] P. Patil, A. Hakiri, Y. Barve, and A. Gokhale, "Enabling software-defined networking for wireless mesh networks in smart environments," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, Oct 2016, pp. 153–157.

[32] R. H. Khan and J. Y. Khan, "A comprehensive review of the application characteristics and traffic requirements of a smart grid communications network," *Computer Networks*, vol. 57, no. 3, pp. 825 – 845, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128612003751

[33] M. Hoefling, F. Heimgaertner, M. Menth, and H. Bontius, "Traffic estimation of the powermatcher application for demand supply matching in smart grids," in *2015 International Conference and Workshops on Networked Systems (NetSys)*, March 2015, pp. 1–6.

[34] S. Tonyali, R. Munoz, K. Akkaya, and U. Ozgur, "A realistic performance evaluation of privacy-preserving protocols for smart grid ami networks," *Journal of Network and Computer Applications*, vol. 119, pp. 24 – 41, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804518302194

[35] B. Heller, "Reproducible network research with high-fidelity emulation," Ph.D. dissertation, 2013, submitted to the Department of Computer Science, Stanford University.

[36] B. Lantz, "Mininet: An Instant Virtual Network on your Laptop (or other PC)," 2018, [Accessed 10-05-2019]. [Online]. Available: http://mininet.org/

[37] R. dos Reis Fontes and C. R. E. Rothenberg, "Mininet-WiFi: Emulator for Software-Defined Wireless Networks," [Accessed 10-05-2019]. [Online]. Available: https://github.com/intrig-unicamp/mininet-wifi

[38] G. Combs, "Wireshark," https://www.wireshark.org/, [Online; Accessed 17-06-2019].

[39] N. S. Nafi, K. Ahmed, M. A. Gregory, and M. Datta, "Software defined neighborhood area network for smart grid applications," *Future Generation Computer Systems*, vol. 79, pp. 500 – 513, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17311007

[40] P. Hintjens, "ZeroMQ - the guide," http://zguide.zeromq.org/, 2012, [Online; accessed 28-05-2019].

[41] ——, "ZeroMQ Realtime Exchange Protocol," https://rfc.zeromq.org/spec:37/ZMTP/, [Online; Accessed 14-06-2019].

[42] Jon Dugan, Seth Elliott, Bruce Mah, Jeff Poskanzer and Kaustubh Prabhu, "iPerf - The ultimate speed test tool for TCP, UDP and SCTP," https://iperf.fr/, [Online; Accessed 17-06-2019].

[43] *Powerwall 2 Datasheet*, Tesla, 2019. [Online]. Available: https://www.tesla.com/sites/default/files/pdfs/powerwall/Powerwall%202_AC_Datasheet_en_northamerica.pdf