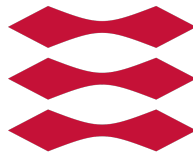


DTU



AGGREGATOR SYSTEM

by

Rosa Llorca Navarro s161340

Dominik Maszczyk s163601

Mirza Nuhic s151891

Ángel Post Vinuesa s155531

INTELLIGENT SYSTEMS

DTU

Electrical engineering

December 2016

Contents

List of Figures	iv
1 Introduction and problem statement	1
2 Concept description	3
2.1 Background	3
2.2 Project context	4
2.3 Aggregator system tasks	7
2.4 Architecture	8
3 Methods and solution	10
3.1 Communication and aggregation	10
3.1.1 Overview	10
3.1.2 Implementation	11
3.1.2.1 Dynamic, local network	11
3.1.2.2 Assumptions	12
3.1.2.3 Aggregator system in wide area network	12
Initialisation	12
Operation	13
3.1.2.4 Fault tolerance	16
3.2 Large Data	17
3.2.1 Data Processing	17
3.2.2 Simulation	20
3.2.3 Machine learning	21
3.3 House Controller and flexibility	24
3.3.1 House Controller	24
3.3.2 The Flexibility Algorithm	25
4 Test cases	31
5 Conclusions and future work	38
5.1 Conclusions	38
5.2 Future work	39
A Contributions	40

B Figures**41**

List of Figures

2.1	TSO procedure: quantities.	5
2.2	TSO timing	6
2.3	Functional Groups	7
2.4	Architecture	8
3.1	Communication overview	11
3.2	Initial broadcasting header and connection established based on it	11
3.3	Initial time syncing propagation invoked by grid	12
3.4	Request propagation	14
3.5	Estimating maximal delta power by a) basic condition b) used algorithm .	15
3.6	Real measurement of the insolation	18
3.7	Insolation after filtering the data	18
3.8	Measured outside temperature	19
3.9	Wind speed generator output	19
3.10	Simulation of the average temperature	20
3.11	Variables that affect the house average temperature	20
3.12	Yreg when the slope is positive	22
3.13	Yreg when the slope is positive	23
3.14	State Chart of the House Controller	25
3.15	Situation when we are receiving request while the system is in a regulating state	27
3.16	Situation when we are receiving the request while the system is in normal state and expecting activation request	28
3.17	Sequence Diagram for Determining Flexibility	29
4.1	House 1 average temp	32
4.2	House 2 average temp	33
4.3	House 2 average temp	33
4.4	34
4.5	35
4.6	36
4.7	Four Flexhouse Simulators running simultaneously	37
B.1	Communication sequence in local network - dynamic solution	41
B.2	Initial time syncing propagation invoked by a) grid b) aggregator c) house controller	42

Chapter 1

Introduction and problem statement

In markets where wind and solar power are starting to approach a significant portion of total grid power, a new challenge has appeared of whether green power and grid stability are compatible, and supply and demand of energy have to match at any given time.

When a power system is operating in a stable state, it means among other things that the power balance is at the equilibrium point, the consumption matches the generation. The energy generated by the wind turbines or solar cells can only be predicted and controlled to some extent.

In the past, we had almost complete control over generation, and almost no control over consumption. With renewable resources, we can have some predictability in the generation, but much less than we had with conventional generating units. So, we have to turn to the demand.

The idea behind the demand response is that we plan for generation forecast errors, and contract a certain amount of fast and controllable resources, in order to preserve the balance in the network by increasing or decreasing consumption.

As electrical heating and cooling loads consume a large amount of power, the resources that were investigated in this project are the electric heating systems in five residential homes. The idea is to develop an aggregator system that has a task to control a certain number of homes and offer the service of demand response to the transmission system operator (TSO).

The home should subscribe to the aggregator if they want to participate in the process and become part of the system. Based on the information from all the homes participating in the service, the aggregator would offer a flexibility range for a certain period of time (for how long can we guarantee to increase or decrease a certain amount of power).

Chapter 2

Concept description

2.1 Background

A transmission system operator (TSO) is one of the main actors in the electricity. It is responsible for operating, ensuring the maintenance of and, if necessary, developing the transmission network in a given area and, where applicable, its interconnection with other networks, as well as for ensuring the long-term ability of the system to meet reasonable demands for the electricity transmission.

In renewable energies, the TSO continuously calculates a forecast of the expected wind or solar power production for the next couple of hours, in order to be able to allocate enough generation resources and match the demand.

Forecast errors are unavoidable in this process, and the TSO will need to contract a certain amount of controllable resources to keep the grid in balance.

Ideally, the output of these regulation resources will then be controlled so they exactly correspond to the power delta caused by the forecasting error (or other miss-allocations).

In practice, the TSO would open a tender for regulation services on a special type of energy market. Market participants, such as aggregators, can then bid on parts of the tender. Due to the time required for the bidding process, the risk of not finding enough bidders for a particular tender and various other reasons, these market transactions cannot be performed when the demand for regulation has already occurred.

Instead, the TSO estimates the demand for regulation resources including some safety margin well ahead of time and contracts the resources to be on stand-by during a particular time interval.

When needed, the TSO sends an activation signal to the winning bidder which will then be obliged to provide the agreed service by adjusting the power consumption or production of the units under its command, being designed high penalties for non-delivery.

2.2 Project context

An aggregator system is designed as a power system regulation service in order to control the demand, being temporarily increased or decreased to help balance the system.

The system controls five residential houses with a total of eight rooms per house. Doors and windows are not taken in account. Each room is electrically heated and thermostatically controlled. Hysteresis is used in order to avoid continuous switching actions, being the set point and his allowed deviation (comfort band) adjusted by the residents.

The aggregator offers a service of demand response to the transmission system operator (TSO). Furthermore, it offers a publish-subscribe pattern to the houses in order to became part of the system. After being subscribed, the houses will send their flexibility range for a certain period of time to the aggregator every time it queries it.

The TSO is given and the procedure is described below:

1. The aggregator receives a request from the TSO, asking to provide a change of power consumption from the houses under his control. The request consists of:
 - Two timestamps (relative to simulation start) marking the beginning and end of the requested service period.
 - The maximum permissible delay from the activation signal to the activation of the service.
 - The minimum delivery duration once the service has been activated. The service will only be activated once per service period.
 - An indication whether power consumption should be increased or decreased (down- or up-regulation).
2. The aggregator responds to the request with the size of the power delta (magnitude of the regulation service) it can guarantee during the service period.
 This should be a positive value if a power increase has been requested, or a negative value if a power decrease has been requested.
 If the aggregator cannot comply with the request, it should return a size of zero.

3. It is assumed that the TSO makes a contract with the aggregator every time a size other than zero is being returned. The TSO is then free to send an activation signal within the agreed period.

In the following figure the relationship between the different quantities is described.

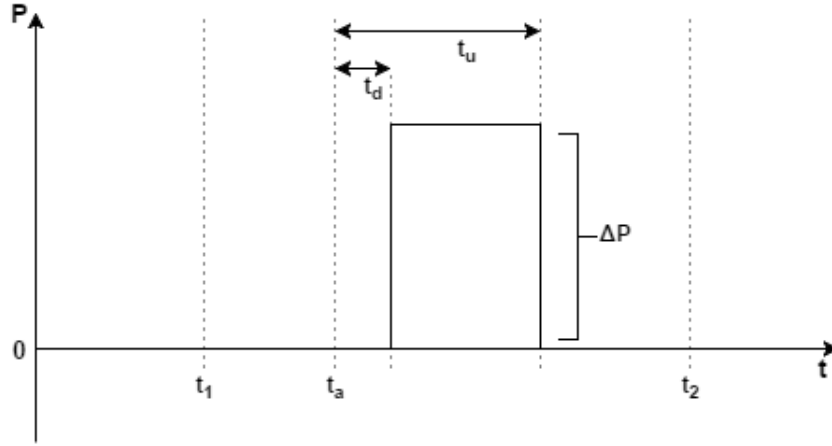


FIGURE 2.1: TSO procedure: quantities.

Being:

- t_1 : Beginning of the contract period.
- t_2 : End of the contract period.
- t_a : Time at which activation signal is received.
- t_d : Maximum permissible activation delay.
- t_u : Minimum duration of service delivery.
- ΔP : Power delta (magnitude of service).

Concerning the timing:

The TSO has a 3-minute cycle of successive contracting periods. Two minutes into the currently running cycle, it negotiates a contract with the aggregator for the next cycle, which starts a minute later (in the end of the current cycle).

The TSO can send an activation signal to the aggregator in order to increase or decrease the power consumption of the system in any time of each contract period.

In the next figure it is shown how the TSO timing works:

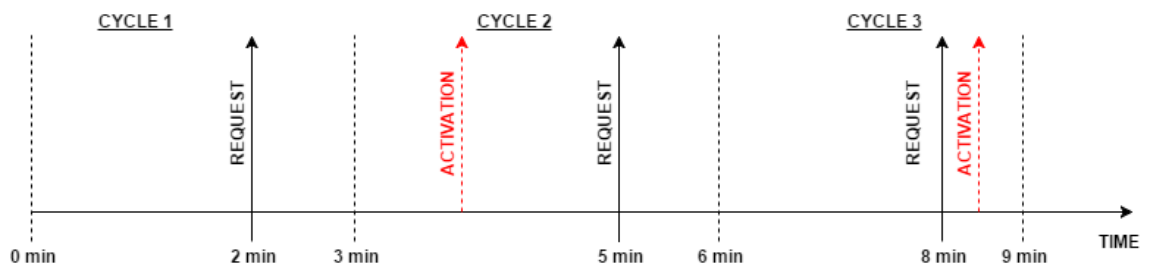


FIGURE 2.2: TSO timing

2.3 Aggregator system tasks

In order to solve the given project, the aggregator system can be analysed in terms of four task types:

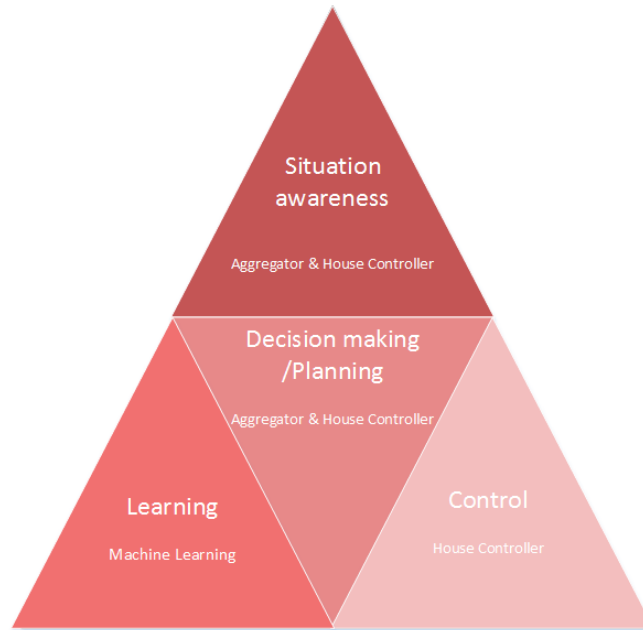


FIGURE 2.3: Functional Groups

- Situation awareness: Aggregator has to be aware of the number of houses that is controlling and the different capabilities. Furthermore, the house controller have to observe and interpret the information from the different sensors of the houses.
- Decision making/planning: Aggregator and house controller use this deliberation task to identify the course of action when it is needed to increase, decrease or maintain the consumption in the houses.
- Control: This task type is used by the house controller in order to regulate the temperature of the house activating or deactivating the heaters of the rooms.
- Learning: Machine learning is used to predict how temperature varies in each house depending on the external temperature.

2.4 Architecture

In the next figure it can be seen how the system has been designed:

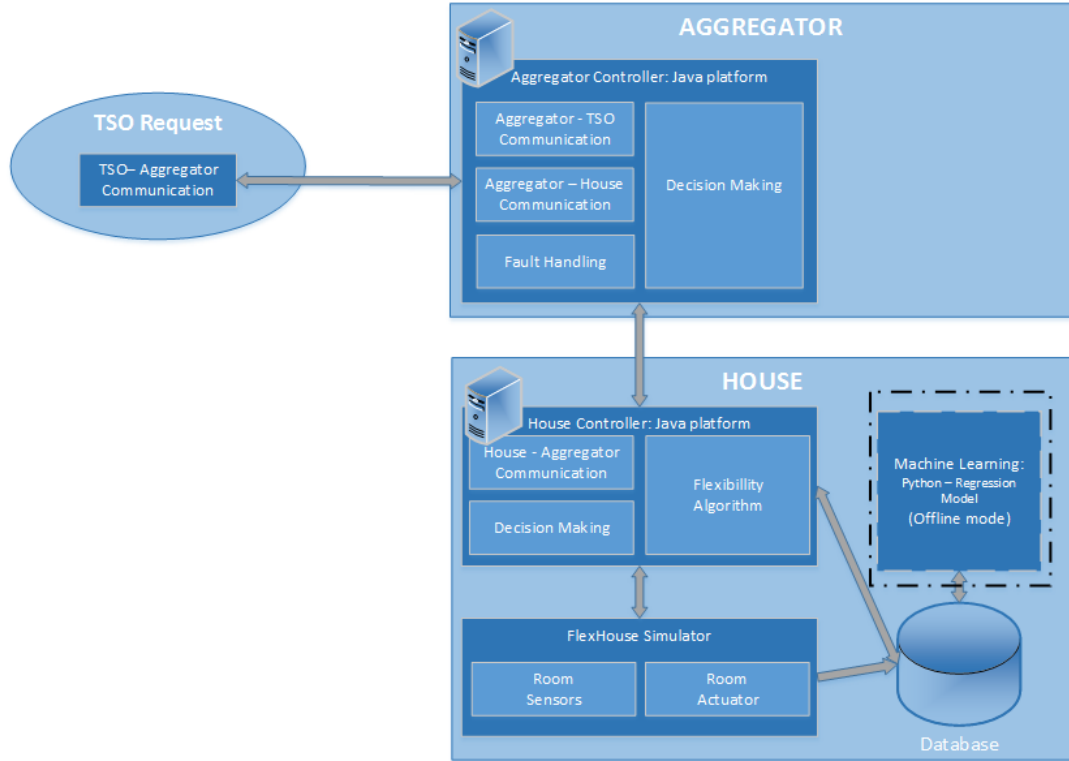


FIGURE 2.4: Architecture

- **Aggregator controller:** Aggregator controller has been implemented in JAVA. It includes TSO-aggregator and aggregator-house controller communication. Furthermore, it has a decision making algorithm where the aggregator controls which houses are going to be used in order to fulfill the regulation order from the TSO. Finally, it manage fault handling like using ACK controlling in order to avoid possible faults in the system.
- **House controller:** House controller has been also implemented in JAVA. It is composed by the house-aggregator communication, the flexibility algorithm that calculates the flexibility available from the house for the next cycle, and a decision making part, where the house controller controls the actuators depending on the temperature settings and the orders received when an activation from the aggregator is required.

- FlexHouse simulator: a temperature simulator of one house used to test the system and acquire data.
- Machine learning: Machine learning has been implemented in Python using regression. It is used in order to predict temperature variations during the day depending on external temperature.

In the next section, each of the components of the system are described with more details.

Chapter 3

Methods and solution

3.1 Communication and aggregation

3.1.1 Overview

To satisfy our goal we developed a system based on three blocks with communication in between of them as follows:

Transmission System Operator - as previously mentioned, TSO makes requests for service of lowering or increasing power usage in our system. All the messages are sent by synchronous calls to grid getting in return acknowledge or exception message. Moreover, the communication is bi-directional, TSO has declared message listener so it can response for certain messages from the aggregator.

Aggregator - is responsible for communication with grid and aggregating messages from houses. Because the connection with houses is not stable and the network addresses are not known/dynamic the communication need be made in a different way. Taking additionally into account number of houses in the network which may get up to few thousands we decide to use a publish-subscribe messaging pattern for sending messages from the aggregator to houses. Responses, however, are sent by standard synchronous calls.

Houses - subscribe to the topic provided by aggregator and respond by synchronous calls.

For more clear picture system is also represented by figure 3.1

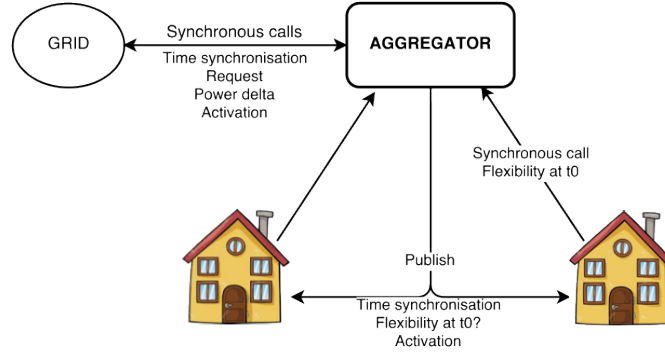


FIGURE 3.1: Communication overview

3.1.2 Implementation

3.1.2.1 Dynamic, local network

Our first idea was to build a network without any predefined addresses. To implement this approach we develop protocol relying mostly on broadcast messages in local network¹. In our protocol, we assume that TSO and aggregator have to periodically broadcasting the message containing their name, address and the port on which communication can be settled. This messages can be interpreted by the aggregator to set connection with TSO, as well as connecting with aggregator Publish server from home perspective. Exact message sequence diagram can be found in the appendixB.1 whereas the basic is operation can be seen in the following figure 3.2.

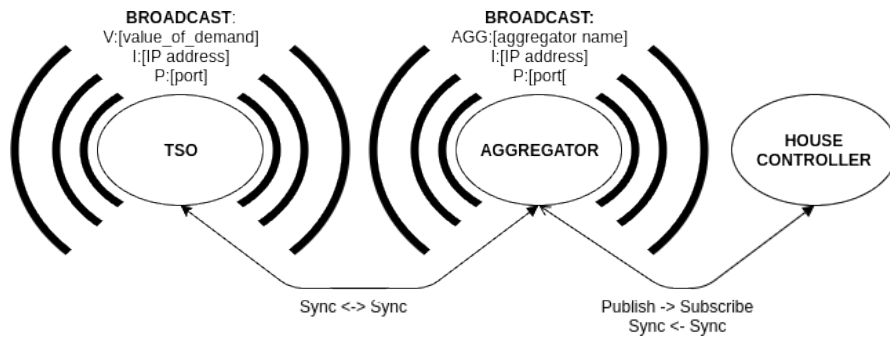


FIGURE 3.2: Initial broadcasting header and connection established based on it

This solution simplifies the network configuration problem to the point where the device needs to just be connected to its local network (plug and play). Moreover, thank its flexibility can work in a network with static as well as dynamic addressing.

¹We assumed that for security and sustainability reasons the best solution would be to connect the devices to a local network.

Although the advantages of this system, it's not used in practice. To implement it the "local" network would need to build the way it covers whole distributed system, which is not cost efficient solution.

3.1.2.2 Assumptions

In following sections we assumed that our system needs to work in wide area network (WAN). Moreover, the aggregator and TSO have a reliable Internet connection and static, known IP addresses.

3.1.2.3 Aggregator system in wide area network

Since the implementation of aggregator system in the local network is not practical in our project we focused on providing system working in WAN. The whole implementation was written in JAVA providing easy to implement and extend the system for communication and define aggregator logic. All the important values were clearly declared in the top of each class and the additional interface was provided to change default values upon calling each part from the terminal.

Initialisation Due to the fact that our communication schematic is based on relative time to the initialisation of TSO upon a start of each component time synchronisation messages are exchanged (figure 3.3).

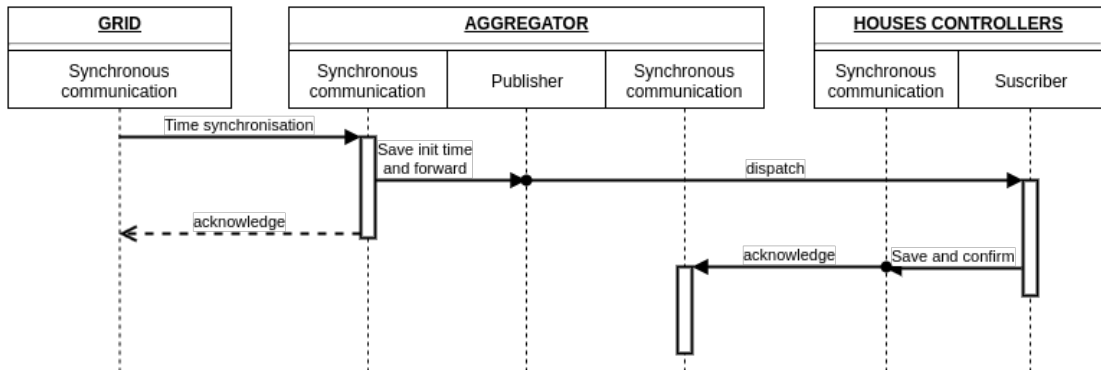


FIGURE 3.3: Initial time syncing propagation invoked by grid

This and all additional cases have been shown in the appendix B.2.

Moreover, each component is going through its initialisation process as follows:

TSO

1. Defines initialisation time equal to the local time upon start.
2. Try up to the success to establish connection with aggregator.
3. Establish messages listener for calls from aggregator.
4. Finally send initial time to aggregator

Aggregator

1. Prepare a list to store flexibility of all active houses along with their flexibility.
2. Set server/messages listeners for calls coming from TSO and houses.
3. Establish a connection with a grid.
4. Set server for publishing messages to houses.
5. Send request to grid for initialisation time.
6. Upon response from grid set initial time and publish it to houses.

House

1. Declare random house name.
2. Establish a synchronous connection with the aggregator.
3. Subscribe to the aggregator.
4. Run the home controller.

The initial messages were designed in the way that system components can be run in random order as well as be correctly initialised after reset.

Operation After the initialization communication is in an idle state waiting for the request or activation from TSO. At the point TSO asks for service request is formatted and simply forwarded to the aggregator. In the aggregator requested service is recognised and served in the following way:

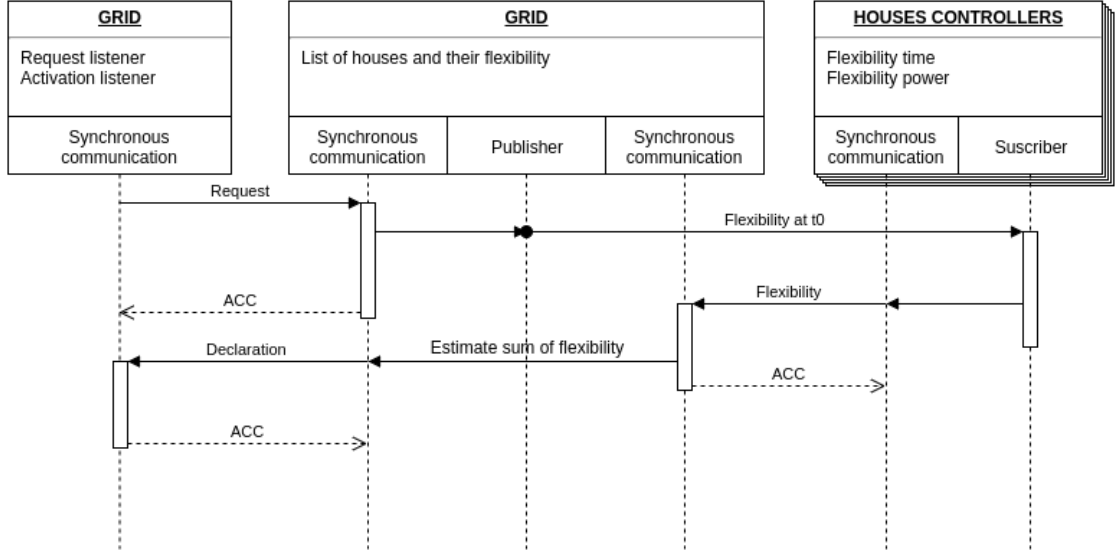


FIGURE 3.4: Request propagation

Request for regulation (contracting) when aggregator receive a request for this service list of active subscribers/houses is dropped, what is directly followed by publishing command, to houses, to declare their flexibility in requested start time. At this point, the thread is stopped for 1s providing enough time for houses to respond by synchronous calls. Each response is stored to the list of active subscribers along with values corresponding to flexibility (time and power). Based on received responses aggregator calculates maximum power Δ^2 which the system is the most likely able to achieve and the result is returned to TSO.

The algorithm used to estimate power delta had been presented in following pseudo-code (corresponding java code can be found in appendix B.1).

```

sort houses which responded by their flexibility_power;
for each (house among houses which responded) {
    if (house flexibility_time > minimal flexibility time requested by
    TSO) delta_power += house flexibility_power;
    else {
        sum_t += home flexibility_time; //sum up houses flexibility times
        to check if sum would be bigger that minimum requested flexibility
        min_p = minimum(min_p, home flexibility_power); //track minimal
        flexibility_power in the group to not overestimate declared
        flexibility
        if (sum_t > minimal flexibility time requested by TSO) { //if
        sequence is giving right minimal time
            delta_power += min_p; //add minimal flexibility_power in this
            time span
        }
    }
}
  
```

²Change in used power through minimal required time

```

        reset sum_t and min_p;
    }
}
}

```

It basically covers two cases:

1. house which has bigger flexibility time than minimal flexibility time requested by grid contributes to delta power.
2. group of houses which together provide flexibility time longer than minimal flexibility time contributes to delta power.

This solution is way more efficient than simply adding a power of houses meeting a requirement to provide flexibility longer than requested minimum flexibility time. Which has been shown in figure 3.5 Moreover, this algorithm can be future upgraded to optimise case of sequencing flexibility in the way it covers possibly just minimal time span and compares many existing options against highest power delta.

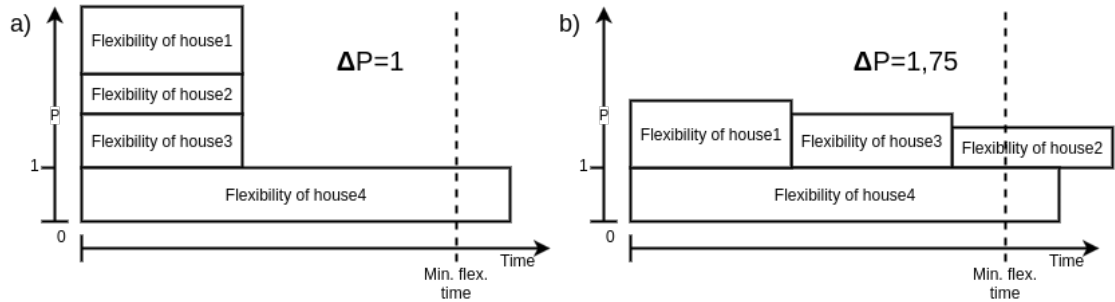


FIGURE 3.5: Estimating maximal delta power by a) basic condition b) used algorithm

When the power delta is successfully estimated aggregator returns its value to TSO and whole system come back into an idle state waiting for activation service request.

Activation As described in chapter 2 where service for request needs respond about flexibility to from a contract, the activation just executes previously declared declaration additionally defining actually needed delta power which might be lower than one offered by aggregator in the contract.

When activation occurs message is immediately forwarded to aggregator which activate houses basing on the sequence defined during power delta calculation. After it, aggregator simply sends acknowledge message to TSO.

3.1.2.4 Fault tolerance

The communication part, as well as whole controlling algorithm, was develop with taking into consideration error handling. The system widely benefits from an idea of sending acknowledge/not acknowledge messages. Most of the synchronous calls would receive in return "ACC" which stands for acknowledging or "EXC" if the message was corrupted or refused. Whereas message is considered corrupted if it does not contain declared function or number of arguments is not proper. In a case of lack of acknowledge non-crucial messages call would be repeated after a random time of 1 to 2 seconds (to reduce a probability that same requests could collide as if they would repeat in constant time) up to 10 times and dropped. On the other hand, the operations which are crucial for further system procedure like setting communication server, receiving initial time etc. are repeated until success. Moreover, in our approach, the problem of subscribers which are not longer present in a network was solved from algorithmic level. All the subscribers confirm their presence by sending a response to the command raised by the publisher.

3.2 Large Data

The aim of this section is to generate data from the simulator in order to obtain the parameters needed for the flexibility algorithm. The data processing and the machine learning are included here.

3.2.1 Data Processing

The simulator used in the flexhouse is the generator data. The first step is to reduce the simulation time, the file `all.csv` used in the simulator contains the temperature from outside, the insolation and the status of the heaters. This data is measured during 24 hours, that is 24 hours of simulation are needed to simulate all the status during the day. To decrease this simulation-time and still having almost all the outside temperature and insolation possibilities that can occur during the day, the large data has been processed using some commands in the terminal:

```
awk 'NR==1 || NR %10 == 0' all.csv > final.csv
```

With that command, a new file has been created (`final.csv`) which contains every 10 data that was in the file `all.csv`. Therefore, the data can be simulated now in less than 3 hours.

Before the simulation, it is important to take into account that the data provided is a real measurement (insolation and outside temperature) and contains some errors which can disturb the simulation. Using python the data has been filtered:

```
final = pd.read_csv('/home/student/Downloads/Exercise10/final.csv', header=0)
out=final[(final['TEMPERATURE_temp1[degC]']<20)&
(final['TEMPERATURE_temp1[degC]']>0)&(final['INSOLATION_irrad1[kW/m2]']<0.25)]
```

It can be seen how is the data before the filter and after filtering it, the measured data for the insolation is plotted in Figure 3.6 , it is easy to see the data that is not desired:

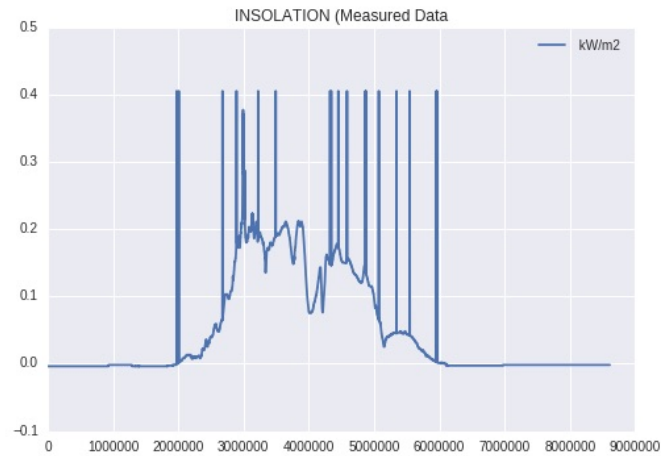


FIGURE 3.6: Real measurement of the insolation

In Figure 3.7 the filter is applied:

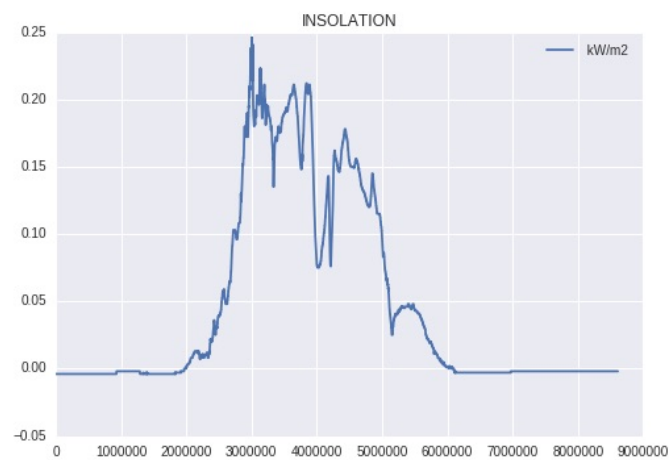


FIGURE 3.7: Insolation after filtering the data

For the outside temperature, it is processed in the same way as the insolation. In the Figure 3.8 it can be seen the data before filtering:

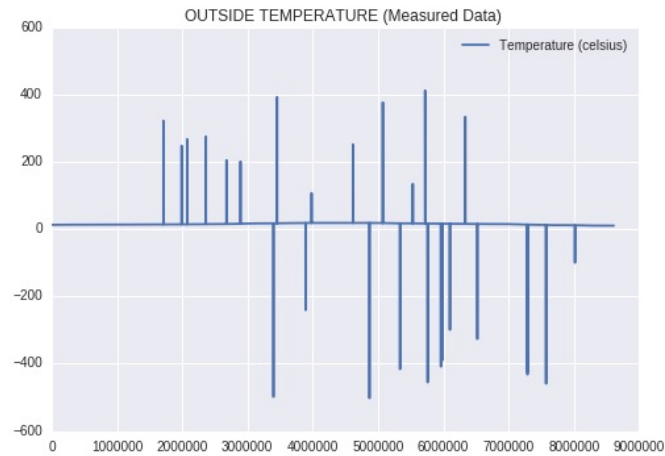


FIGURE 3.8: Measured outside temperature

In Figure 3.9 the filter is applied:

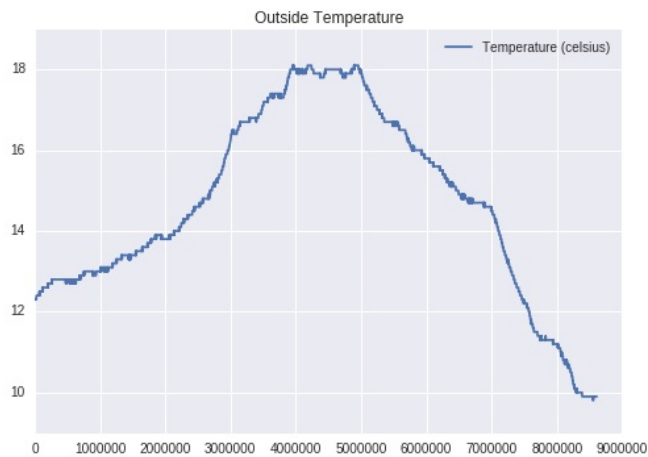


FIGURE 3.9: Wind speed generator output

As it can be seen in the graphs, there is some information that contains some errors of measurement. The new data is saved in a new .csv file, that new file substitutes the old file 'all.csv' used in the simulator.

```
out.to_csv('newfile.csv')
```

3.2.2 Simulation

Once the new file is processed, the simulation can be done. For the simulation it is assumed that the windows are always closed, there is no occupancy, the doors status are not taken into account and it is calculated the average temperature for each house as the sum of every room over the number of rooms. The heaters have only two status: all the heaters on or all the heaters off, there is no independence between them.

The output of the simulation is the average temperature in the house. The java file has been programmed to obtain the average temperature in the house Figure 3.10, this average temperature has a maximum and a minimum point fixed (min=19°C, max=25°C), when the boundaries are reached the status conditions of the heaters change. If the maximum temperature is reached all the heaters are turned off while when the minimum temperature is reached all the heaters are turned on. As can be seen in Figure 3.11, this average temperature depends on three main factors: the insolation, the outside temperature and the state of the heaters. This dependence is important when the linear regression is applied.

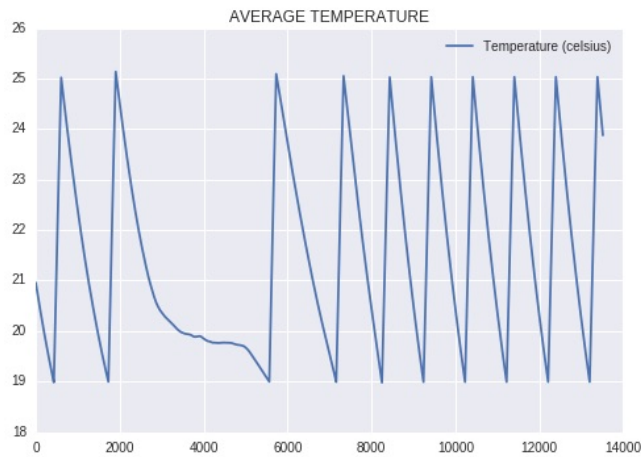


FIGURE 3.10: Simulation of the average temperature

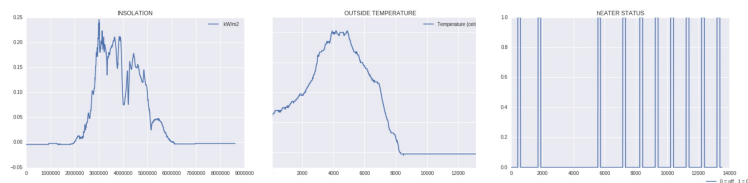


FIGURE 3.11: Variables that affect the house average temperature

3.2.3 Machine learning

When the simulation is done, all the data needed for the learning machine is ready. The learning machine is applied in order to obtain the flexibility in each house. In this project, it is simulated one sort of house, for the other houses the same method can be applied. As the goal is to obtain the flexibility in each house, the first point is to define what is considered flexibility. In this case the flexibility has been considered as the time needed for heating the house until the maximum temperature point, and how long lasts to reach to minimum temperature point when the heaters are turned off.

The learning machine it is solved using the linear regression method, with the linear regression the model is estimated. The simple linear regression consists on one variable that depends on other independent variables, in this case the dependent variable is the average temperature which depends on the other two variables: insolation and outside temperature. But as the data has been adapted in order to have a smaller time simulation, all the data measured during 24 for hours is not available, hence it appears some errors when the linear regression is applied taking into account the insolation, it is why the linear regression has been approximated only with one dependent variable, the average temperature (y) and the outside temperature as the independent variable (x). The linear regression applied looks like the equation (3.1):

$$y = a x \quad (3.1)$$

Adding our parameters to the linear regression that has been chosen the equation (3.2) has been obtained. In this project the variable y is the increment of the average temperature (equation 3.3) (that can be either positive or negative) over the time needed to perform that change (equation).

$$\frac{\Delta T}{\Delta t} = \alpha T_{out} \quad (3.2)$$

$$\Delta T = T(i+1) - T(i) \quad (3.3)$$

$$\Delta t = t(i+1) - t(i) \quad (3.4)$$

It has been implemented in python:

```
time=data['timestamp']
tout=data['temp_average']
T=[0]*len(time)
i=1
while i<len(time):
    T[i-1]=tout[i]-tout[i-1]
    i=i+1

t=[0]*len(time)
j=1
while j<len(time):
    t[j-1]=time[j]-time[j-1]
    j=j+1

Y_reg=[0]*len(t)
i=1
while i<(len(t)-1):
    Y_reg[i]=T[i]/t[i]
    i=i+1
```

Due to the fact that the slope when the house is being heated and the slope when the heaters are turned off is different, there are two different models: one when the heaters are turned on (positive slope) Figure 3.12 and the other when the heaters are turned off (negative slope) 3.13, the aim is to split this different data two obtain the two models. In order to do that, the data from Yreg is classified as Yreg with positive slope (heaters on) and Yreg with negative slope (heaters off)

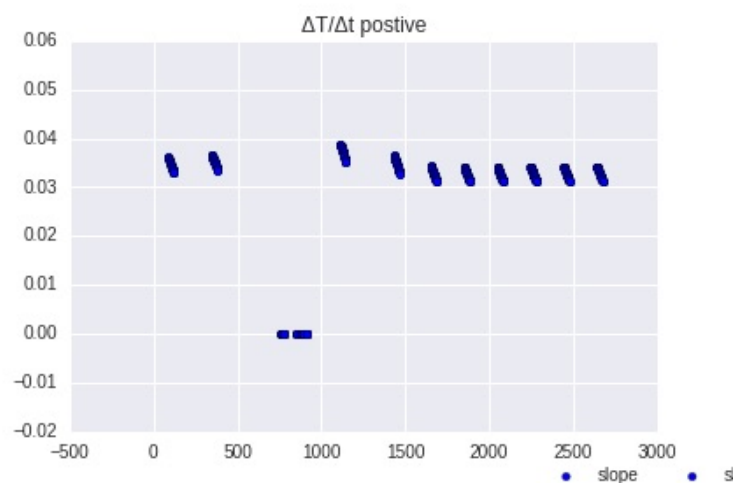


FIGURE 3.12: Yreg when the slope is positive

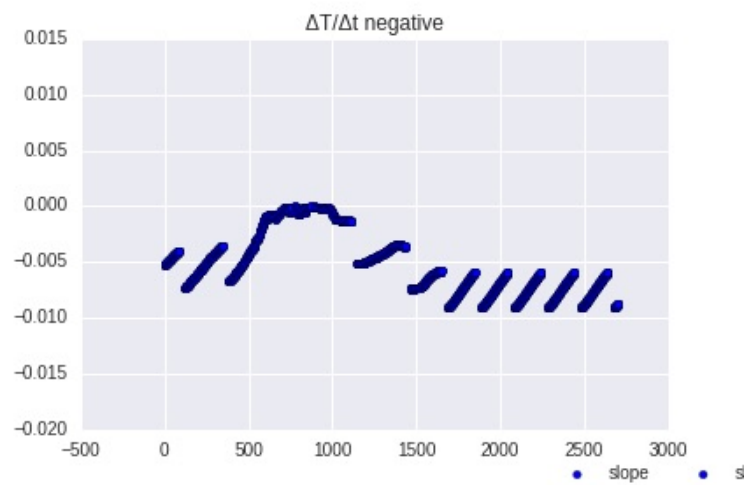


FIGURE 3.13: Yreg when the slope is positive

Applying linear regression the coefficients needed are obtained:

α positive = -0.00257949

α negative = 0.00085741

Once the two different models are obtained, the cross validation is done to estimate the accuracy of the predictive model.

3.3 House Controller and flexibility

3.3.1 House Controller

The residential heating system is controlled by a simple hysteresis thermostat. In order to simplify the process of control and to show the basic idea behind the aggregator system and demand response, we chose to control the average temperature within a house. That means that we are turning on the heaters in each room on or off, depending on the type of the service that is requested ("down" or "up" regulation), and the controlled temperature is the average temperature of all rooms.

The hysteresis thermostat is based on a delay in response, in order to prevent the influence of temperature sensor noise, as well as wear and tear of the device. Instead of a temperature set point, the thermostat is operating in a band around the set point. So, for example, if the temperature is set to 21°C , and the band around the set point is $\pm 0.25^{\circ}\text{C}$, the thermostat will turn on when the temperature is below 20.75°C , and it will turn off when the temperature is above 21.25°C .

The house controller can be in four possible states:

1. Normal operation (within band limits and not regulating);
2. Regulating down (all actuators are on);
3. Regulating up (all actuators are off);
4. Transitional state (regulating and going back to normal operation).

The state chart of the house controller is given in figure 3.14. The controller is initialised when the simulator starts, and it immediately goes to normal operation. The state change from this point can only be initialised by the external agent, namely the aggregator. When the aggregator activates the service, we change the state to UP or DOWN regulation, depending on the TSO request. From this point, there is only one direction for the state change, and it is to a transitional state. The change is activated automatically if one of the following conditions are satisfied:

1. The contract interval is exceeded;
2. The maximum or minimum regulating temperature is exceeded.

From the transitional state, we can go back to regulating state if the service for the next contract is activated. The other possibility is that the temperature is within the thermostat range and the controller will go back to normal operation. Final state of the controller is when the simulator is stopped.

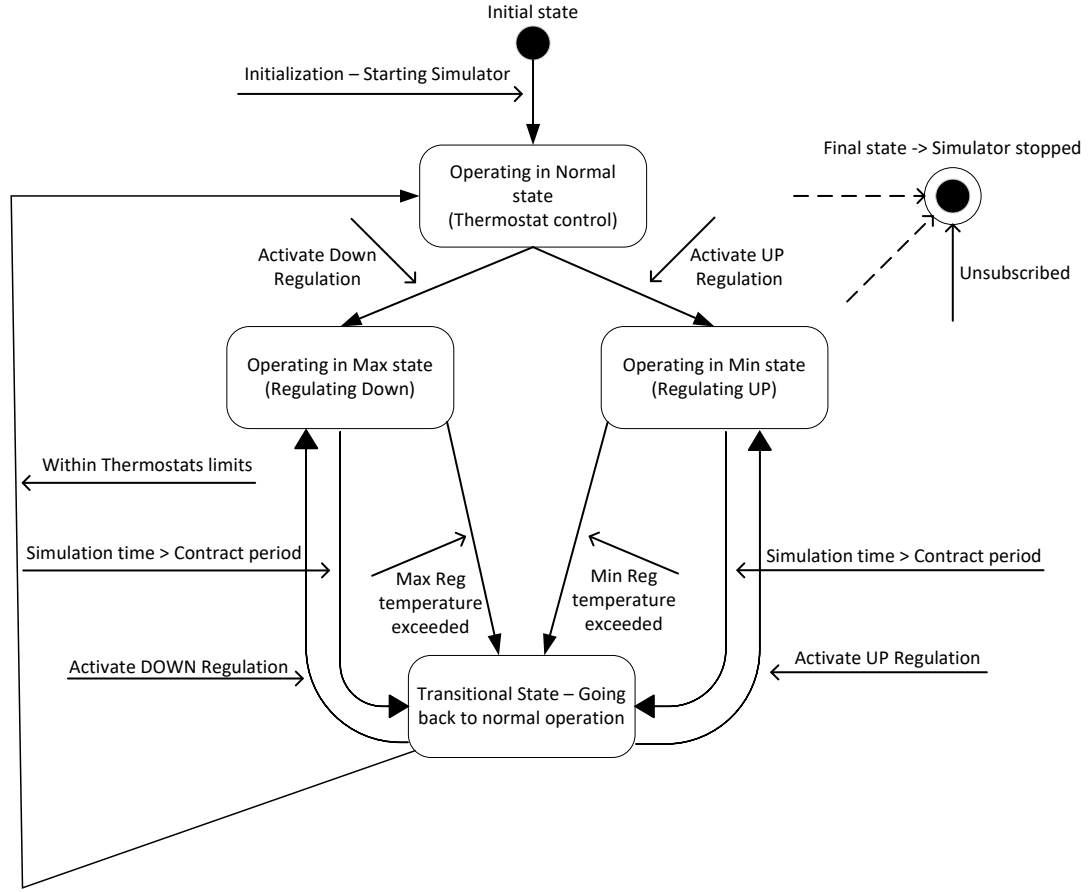


FIGURE 3.14: State Chart of the House Controller

3.3.2 The Flexibility Algorithm

The definition of the flexibility used throughout this report is the time for which a certain consumer (residential heating system) can guarantee an increase or decrease of certain amount of power. This time interval is basically a time that it takes for a house to heat up to maximum temperature set point, or to cool down to minimum temperature set point. A regulation range, namely a range of temperature values between maximum and minimum set points, is defined and determined by the home owner, and we cannot operate outside of this range.

The flexibility of a residential home is determined based on a specific thermal characteristic of a building or a house, as well as the heating system capability. So, in order

to predict the behaviour of inside temperature curve, we used machine learning to determine the slope of temperature drop and rise with time. The basic idea is shown in the following figure.

Since we are dealing with small ranges of temperature, we assumed that the slopes are linear. The equation describing the slope of the temperature rise is given as:

$$\frac{T_{max} - T_{current}}{t_{Tmax} - t_{current}} = \frac{\Delta T_{down}}{\Delta t_{down}} = c_{down} \cdot T_{out} \quad (3.5)$$

where T_{max} and $T_{current}$ are the maximum and current temperature, respectively, the t_{Tmax} and $t_{current}$ are the corresponding times, T_{out} is the measured outside temperature, while the c_{down} is a coefficient that relates the slope of the inside temperature and the current value of the outside temperature. From the equation given in 3.5 we can determine the time we can offer the "down" regulation service as:

$$\Delta t_{down} = \frac{\Delta T_{down}}{c_{down} \cdot T_{out}} \quad (3.6)$$

Similar, for the "up" regulation we have:

$$\begin{aligned} \frac{T_{current} - T_{min}}{t_{current} - t_{Tmin}} &= \frac{\Delta T_{up}}{\Delta t_{up}} = c_{up} \cdot T_{out} \\ \Delta t_{up} &= \frac{\Delta T_{up}}{c_{up} \cdot T_{out}} \end{aligned} \quad (3.7)$$

Now that we defined the procedure for calculating the flexibility time, we can consider the cases that we might encounter in real time operation. As previously explained, the contract period, the contract start time, the contract end time, as well as the flexibility request time, are all fixed and defined by the transmission system operator (TSO). The only parameter that is random is the time when the activation of the service happens. This uncertainty is forcing us to consider different cases with different controller states. Namely, the aggregator will request the flexibility from the residential homes, and within the house controller we have to determine the flexibility for the next contract period with taking into account the current state, and the possible future states of the controller.

First case that we are going to consider is a case when the aggregator is requesting for flexibility for the next contract, while we are already offering some service, which means

that the controller is in the regulating state. The situation is shown graphically in figure 3.15.

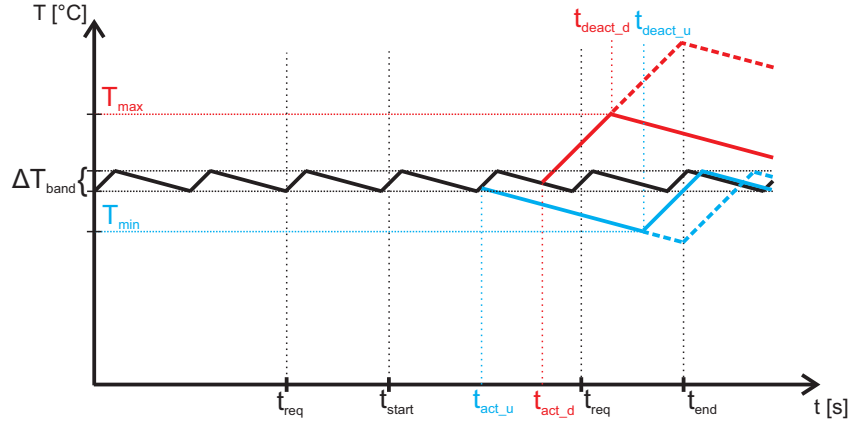


FIGURE 3.15: Situation when we are receiving request while the system is in a regulating state

The notations are as following:

- T_{max} - maximum temperature for the regulating process;
- T_{min} - minimum temperature for the regulating process;
- ΔT_{band} - Hysteresis range of the thermostat;
- t_{req} - time of the request for the next contract;
- t_{start} - start time of the next contract;
- t_{end} - end time of the next contract;
- t_{act_u} - activation time of "up" regulation;
- t_{deact_u} - time when the temperature reaches the T_{min} and the controller goes back to normal operation;
- t_{act_d} - activation time of "down" regulation;
- t_{deact_d} - time when the temperature reaches the T_{max} and the controller goes back to normal operation;

The figure shows four possible cases when a request comes. The full black line is the normal operation of the controller, the red line represents the "up" regulation, while the blue line represents the "down" regulation.

For "down" regulation we have two possible cases. The full blue line represents a case where a minimum regulating temperature is reached before the current contract period

ends. So, in order to calculate the flexibility, we have to determine the temperature at the start of the next contract. First we calculate the time that it takes the temperature to reach a minimum regulating temperature (t_{deact_u}), and this is given in 3.7. Now, since we know the temperature in that point (T_{min}), and the time t_{end} , we can calculate the temperature at time t_{end} .

$$\begin{aligned} \Delta t &= t_{end} - (t_{act_u} + \Delta t_{up}) \\ \frac{T - T_{min}}{\Delta t} &= c_{down} \cdot T_{out} \Rightarrow T = T_{min} + c_{down} \cdot T_{out} \cdot \Delta t \end{aligned} \quad (3.8)$$

So, this temperature will be the initial value for the flexibility calculation for the next contract (equations given in 3.6 and 3.7).

In a case where the minimum temperature is not reached before the current contract ends, the following equations apply:

$$\begin{aligned} \Delta t &= t_{end} - t_{act_u} \\ \frac{T_{setpoint} - T}{\Delta t} &= c_{up} \cdot T_{out} \Rightarrow T = T_{setpoint} - c_{up} \cdot T_{out} \cdot \Delta t \end{aligned} \quad (3.9)$$

Based on this temperature, the flexibility for the next contract is determined. Same logic can be applied when the current regulation is "down". The remaining four cases are when the service is not yet activated, and we receive a request for flexibility for the next contract. This scenario is a little bit challenging, because we are not aware when the activation of the service for the current contract will happen. The situation is shown in figure 3.16.

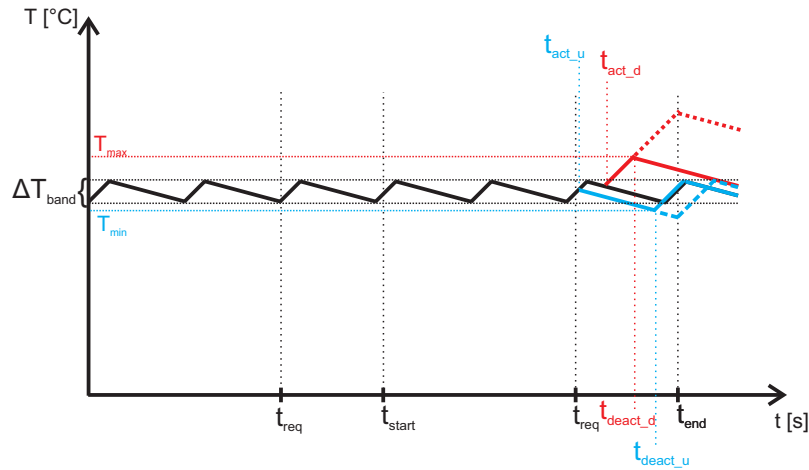


FIGURE 3.16: Situation when we are receiving the request while the system is in normal state and expecting activation request

For "up" regulation we have two possible cases, just as in the previous figure. But, since we do not know the activation time of the service or will it happen at all, we have to assume the most extreme possible outcome in order to be able to guarantee a certain service. The largest deviation in temperature will happen if the activation for the current contract happens immediately after the request for the next contract. The most severe case is if the maximum is not reached before the contract ends (dashed blue line), in which case we will have the largest deviation of temperature from the set point. The temperature at t_{end} can be calculated as per equation given in 3.9, where Δt is given as:

$$\Delta t = t_{end} - t_{req} \quad (3.10)$$

The logic is quite similar for the other three cases, as we explained for the previous figure.

The sequence diagram of the java function that determines the flexibility time is given in figure 3.17.

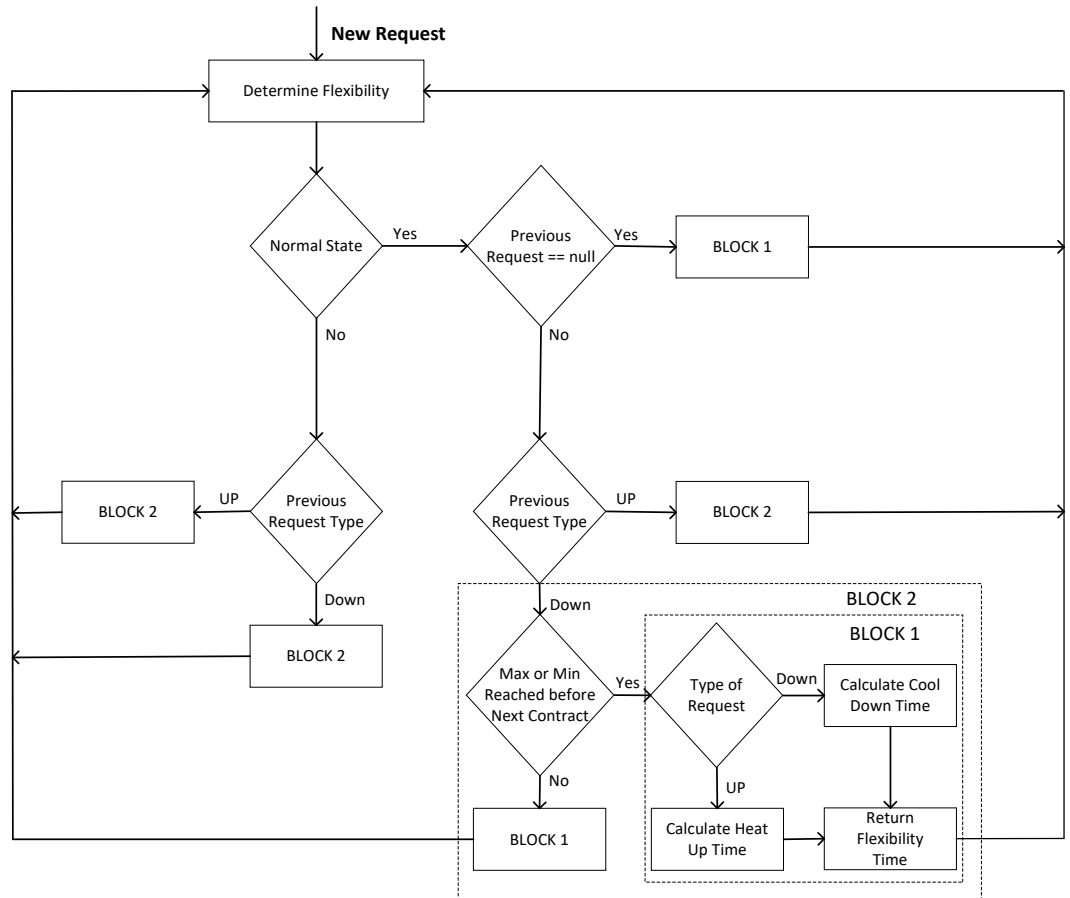


FIGURE 3.17: Sequence Diagram for Determining Flexibility

The sequence diagram shows the procedure for determining the flexibility. The algorithm is activated when the request for service has been made. Upon activation the current state of the controller is determined. If the controller is in normal operation, and there is no active requests, the flexibility is determined as per equations given in 3.6 and 3.7. Otherwise, we have a case where the activation of the service for the current contract has not happened yet, so we refer to situation described by figure 3.16. The logic is similar for all other cases. The boxes marked with notations "Block 1" and "Block 2" are just there in order not to repeat the same procedure describing the same thing.

Chapter 4

Test cases

In order to test and validate the integrated aggregator system, we have prepared a simple case where we offer demand response based on five residential homes. To test the system, the next settings are taken:

- Provided FlexHouse simulator is used for each of the houses.
- Five houses are subscribed to the aggregator through the publish-subscribe service.
- A deviation of 0.25 degrees and a setpoint of temperature in degrees in each house is set.
- Maximum and minimum temperature is defined by the consumers, and we don't have control over this.
- TSO timings are set as in the Figure 2.2, while the activation is sent at a random time within a contract cycle.
- Temperature graphs are recorded from the simulator and showed in this section.
- Screen shots of the console are showed as a result of the test in order to show what is happening in each of the components.

An example of settings of one house are given as:

1. Set point temp: 21°C and hysteresis deviation: $\pm 0.25^{\circ}$;
2. Max temp: 19°C , and min temp: 24°C

In order to show a clearer picture of what's happening with the average temperature of the house (and consequently the power consumption), we extracted the values from the FlexHouse Simulator, and plotted them in MATLAB. The response is shown in figure 4.1.

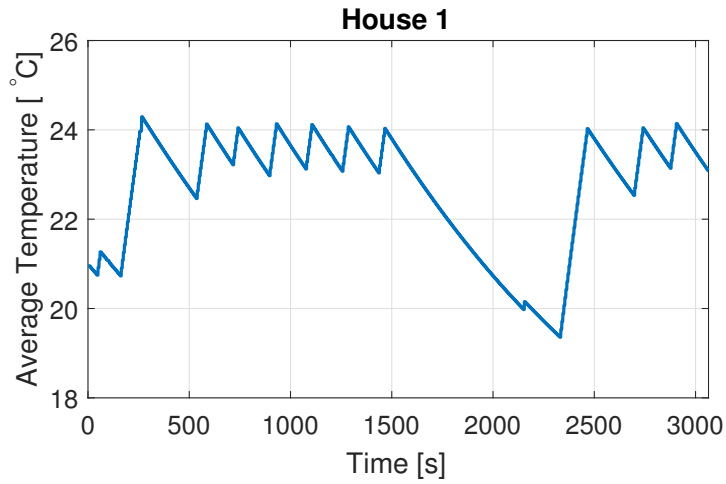


FIGURE 4.1: House 1 average temp

The initial temperature band is the hysteresis control of the thermostat, because when the controller is initialised for the first time, it is in normal operation. From now on the aggregator is sending the activation signals to the house.

For House 2, the values are the following:

1. Set point temp: 21°C , and the hysteresis deviation is: $\pm 0.25^{\circ}$;
2. Min temp: 20°C , and max temp: 23°C

The response for this house is shown in figure 4.2.

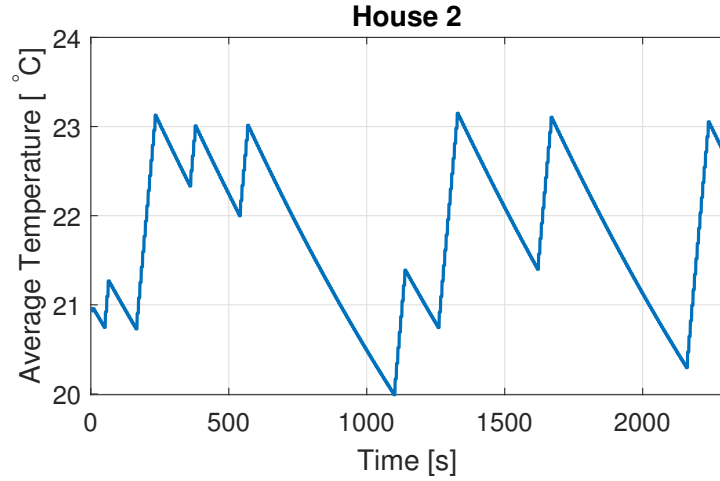


FIGURE 4.2: House 2 average temp

In the following graph (figure 4.3) we can see the same graph as in 4.2 taken directly from the FlexHouse Simulator.

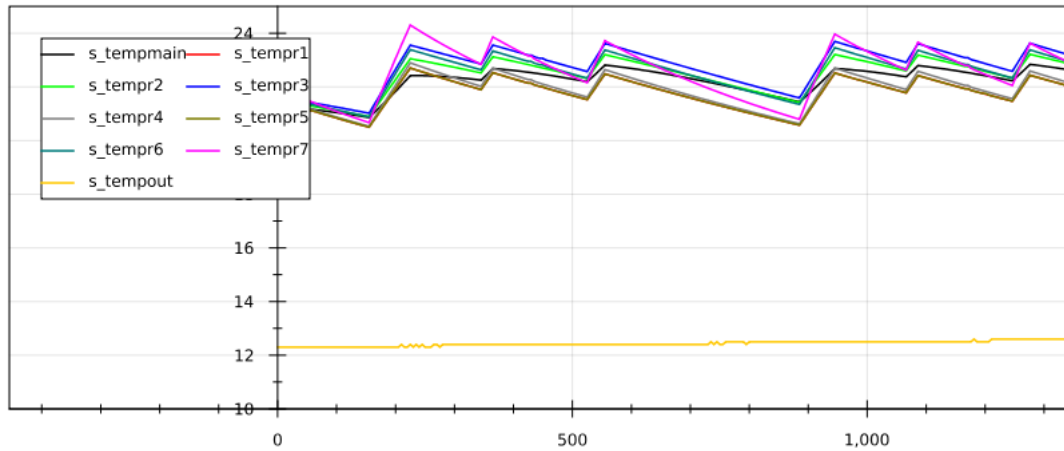
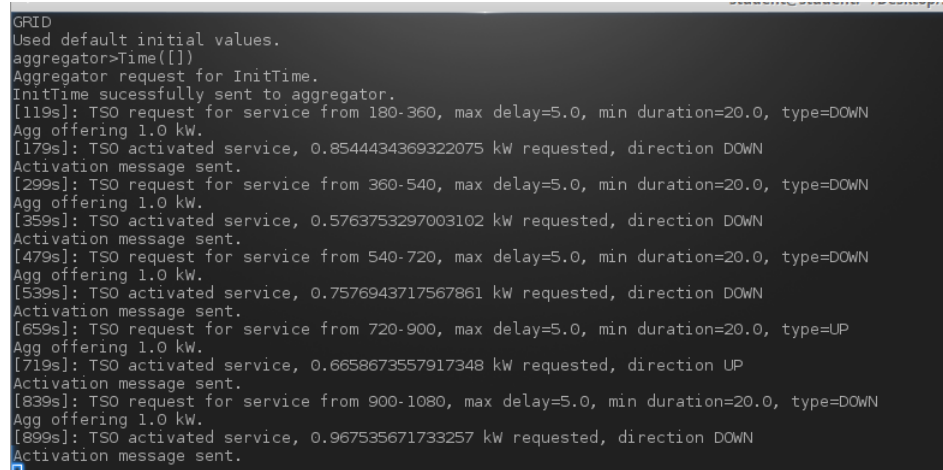


FIGURE 4.3: House 2 average temp

Here it can be seen how temperature varies during the time of simulation. First, the temperature goes to the set point, and it is into the deviation range chosen. After, when the activation takes part requiring to decrease the consumption, the heaters are switched off and the temperature goes down until the defined minimum temperature (19 degrees) Then, temperatures goes to the normal state again.

From the TSO console it can be seen how the system works:



```
GRID
Used default initial values.
aggregator>Time([])
Aggregator request for InitTime.
InitTime successfully sent to aggregator.
[119s]: TSO request for service from 180-360, max delay=5.0, min duration=20.0, type=DOWN
Agg offering 1.0 kW.
[179s]: TSO activated service, 0.8544434369322075 kW requested, direction DOWN
Activation message sent.
[299s]: TSO request for service from 360-540, max delay=5.0, min duration=20.0, type=DOWN
Agg offering 1.0 kW.
[359s]: TSO activated service, 0.5763753297003102 kW requested, direction DOWN
Activation message sent.
[479s]: TSO request for service from 540-720, max delay=5.0, min duration=20.0, type=DOWN
Agg offering 1.0 kW.
[539s]: TSO activated service, 0.7576943717567861 kW requested, direction DOWN
Activation message sent.
[659s]: TSO request for service from 720-900, max delay=5.0, min duration=20.0, type=UP
Agg offering 1.0 kW.
[719s]: TSO activated service, 0.6658673557917348 kW requested, direction UP
Activation message sent.
[839s]: TSO request for service from 900-1080, max delay=5.0, min duration=20.0, type=DOWN
Agg offering 1.0 kW.
[899s]: TSO activated service, 0.967535671733257 kW requested, direction DOWN
Activation message sent.
```

FIGURE 4.4

The system is initialized, and InitTime is setted up in order to control the different cycles. After two minutes, there is the TSO request with the times requirements explained in the previous sections. Then, after receiving the flexibility, an activation signal is sent in the second 179, asking for 0,8544kW to increase the consumption (type DOWN). Then again, after 2 minutes of the second cycle, a request is sent in order to know the flexibility for the next period, and afterwards another activation is sent.

Now, aggregator console is shown in the next figure:

```

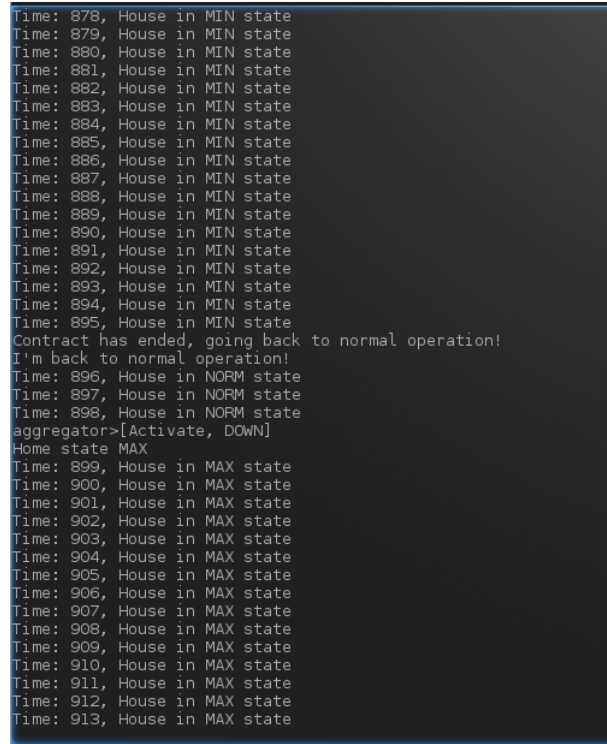
AGGREGATOR
Used default initial values.
Calls listeners set
Connected to grid.
Publisher server started.
Asking grid for InitTime.
InitTime set to 1480564925680.
Sending initTime to houses
Time published.
Subscriber 'house0.7813559123917627' joined for topic 'Flexibility. Sending the InitTime.
house>Time([ACC])
grid>Request([180, 360, 5.0, 20.0, DOWN])
Forwarding request to houses.
Waiting one second for flexibility informations from houses.
house>FlexT0([house0.7813559123917627, 50.8130081300813, 1.0])
1 houses responded.
Home name : flexibility time : flexibility power
house0.7813559123917627 : 50.8130081300813 : 1.0
grid>Activation([DOWN, 0.8544434369322075])
house>Activate([ACC])
grid>Request([360, 540, 5.0, 20.0, DOWN])
Forwarding request to houses.
Waiting one second for flexibility informations from houses.
house>FlexT0([house0.7813559123917627, 15.665518166959453, 1.0])
1 houses responded.
Home name : flexibility time : flexibility power
house0.7813559123917627 : 15.665518166959453 : 1.0
grid>Activation([DOWN, 0.5763753297003102])
house>Activate([ACC])
grid>Request([540, 720, 5.0, 20.0, DOWN])
Forwarding request to houses.
Waiting one second for flexibility informations from houses.
house>FlexT0([house0.7813559123917627, 25.512735258352702, 1.0])
1 houses responded.
Home name : flexibility time : flexibility power
house0.7813559123917627 : 25.512735258352702 : 1.0
grid>Activation([DOWN, 0.7576943717567861])
house>Activate([ACC])
grid>Request([720, 900, 5.0, 20.0, UP])
Forwarding request to houses.

```

FIGURE 4.5

First, aggregator is initialized, connected to the grid and set all the timing needed to be working. Then, the house is subscribed into the aggregator publish-subscribe service and time settings are sent to the house. Afterwards, aggregator receives the request from the TSO, and requires the information to the house controller. After the flexibility is received, an activation signal is sent by the TSO. Aggregator manage the order to the house controller and receives a confirmation from it. Finally, same process is done during the next cycles.

In the house controller console, it can be seen how the house changes into different states depending of the orders received from the aggregator.



```
Time: 878, House in MIN state
Time: 879, House in MIN state
Time: 880, House in MIN state
Time: 881, House in MIN state
Time: 882, House in MIN state
Time: 883, House in MIN state
Time: 884, House in MIN state
Time: 885, House in MIN state
Time: 886, House in MIN state
Time: 887, House in MIN state
Time: 888, House in MIN state
Time: 889, House in MIN state
Time: 890, House in MIN state
Time: 891, House in MIN state
Time: 892, House in MIN state
Time: 893, House in MIN state
Time: 894, House in MIN state
Time: 895, House in MIN state
Contract has ended, going back to normal operation!
I'm back to normal operation!
Time: 896, House in NORM state
Time: 897, House in NORM state
Time: 898, House in NORM state
aggregator>[Activate, DOWN]
Home state MAX
Time: 899, House in MAX state
Time: 900, House in MAX state
Time: 901, House in MAX state
Time: 902, House in MAX state
Time: 903, House in MAX state
Time: 904, House in MAX state
Time: 905, House in MAX state
Time: 906, House in MAX state
Time: 907, House in MAX state
Time: 908, House in MAX state
Time: 909, House in MAX state
Time: 910, House in MAX state
Time: 911, House in MAX state
Time: 912, House in MAX state
Time: 913, House in MAX state
```

FIGURE 4.6

In that case, the house is first maintained in the MIN state, and goes back to the NORMAL state when a new contract period starts. Afterwards, when an activation signal of type DOWN is received, the house goes to the MAX state activating the heaters of the house.

Finally it is shown how the system works with four FlexHouse simulators running at the same time.

```

house_controller.sh
GRID
Used default initial values.
aggregator>Time(175)
Aggregator request for InitTime.
InitTime successfully sent to aggregator.
[175s]: TSO request for service from 180-360, max delay=5.0, min duration=20.0, type=UP
Agg offering 4.0 kw.
[176s]: TSO activated service, 3.69356373889254 kw requested, direction UP
Activation message sent.

house>Time([ACC])
house>Time([ACC])
Subscriber 'house0.30750850590662704' joined for topic 'Flexibility. Sending the InitTime
house>Time([ACC])
house>Time([ACC])
house>Time([ACC])
gridRequest([180, 360, 5.0, 20.0, UP])
forwarding request to houses.
Waiting one second for flexibility informations from houses.
house>FlexT0([house0.6159823677297878, 159.6220150683182, 1.0])
house>FlexT0([house0.33908580267912347, 159.6220150683182, 1.0])
house>FlexT0([house0.30750850590662704, 159.6220150683182, 1.0])
house>FlexT0([house0.04899183407314489, 159.6220150683182, 1.0])
4 houses responded.
Home name : flexibility time : flexibility power
house0.6159823677297878 : 159.6220150683182 : 1.0
house0.33908580267912347 : 159.6220150683182 : 1.0
house0.30750850590662704 : 159.6220150683182 : 1.0
house0.04899183407314489 : 159.6220150683182 : 1.0
grid>Activation([UP, 3.69356373889254])
house>Activate([ACC])
house>Activate([ACC])
house>Activate([ACC])
house>Activate([ACC])

Time: 175, House in NORM state
Time: 176, House in NORM state
Time: 177, House in NORM state
Time: 178, House in NORM state
aggregator>[Activate, UP]
Home state MIN
Time: 179, House in MIN state
Time: 180, House in MIN state
Time: 181, House in MIN state
Time: 182, House in MIN state
Time: 183, House in MIN state
Time: 184, House in MIN state
Time: 185, House in MIN state
Time: 186, House in MIN state
Time: 187, House in MIN state
Time: 188, House in MIN state
Time: 189, House in MIN state
Time: 190, House in MIN state
Time: 191, House in MIN state
Time: 192, House in MIN state
Time: 193, House in MIN state
Time: 194, House in MIN state
Time: 195, House in MIN state
Time: 196, House in MIN state
Time: 197, House in MIN state
Time: 198, House in MIN state
Time: 199, House in MIN state
Time: 200, House in MIN state

Time: 175, House in NORM state
Time: 176, House in NORM state
Time: 177, House in NORM state
Time: 178, House in NORM state
aggregator>[Activate, UP]
Home state MIN
Time: 179, House in MIN state
Time: 180, House in MIN state
Time: 181, House in MIN state
Time: 182, House in MIN state
Time: 183, House in MIN state
Time: 184, House in MIN state
Time: 185, House in MIN state
Time: 186, House in MIN state
Time: 187, House in MIN state
Time: 188, House in MIN state
Time: 189, House in MIN state
Time: 190, House in MIN state
Time: 191, House in MIN state
Time: 192, House in MIN state
Time: 193, House in MIN state
Time: 194, House in MIN state
Time: 195, House in MIN state
Time: 196, House in MIN state
Time: 197, House in MIN state
Time: 198, House in MIN state
Time: 199, House in MIN state
Time: 200, House in MIN state

Time: 175, House in NORM state
Time: 176, House in NORM state
Time: 177, House in NORM state
Time: 178, House in NORM state
aggregator>[Activate, UP]
Home state MIN
Time: 179, House in MIN state
Time: 180, House in MIN state
Time: 181, House in MIN state
Time: 182, House in MIN state
Time: 183, House in MIN state
Time: 184, House in MIN state
Time: 185, House in MIN state
Time: 186, House in MIN state
Time: 187, House in MIN state
Time: 188, House in MIN state
Time: 189, House in MIN state
Time: 190, House in MIN state
Time: 191, House in MIN state
Time: 192, House in MIN state
Time: 193, House in MIN state
Time: 194, House in MIN state
Time: 195, House in MIN state
Time: 196, House in MIN state
Time: 197, House in MIN state
Time: 198, House in MIN state
Time: 199, House in MIN state
Time: 200, House in MIN state

```

FIGURE 4.7: Four Flexhouse Simulators running simultaneously

Similarly to the previous explanations, it can be seen that the system works correctly running either with one or several houses. It can be seen how communication works between TSO and aggregator, and how aggregator is communicated with the different houses.

Chapter 5

Conclusions and future work

5.1 Conclusions

Our project presents successful proof on the concept to balance energy network by using distributed system of cooperating households. We ensured durable communication protocol in between of system components, which is easily scalable to command thousands of households simultaneously. Moreover, we are successfully controlling all the heaters in homes with respect to preset temperature values (possibility of adding the user interface to change temperatures in a home). Based on the network of sensors in home and observation of their values we train our model which over the time is capable of describing house characteristic especially in the manner of house "flexibility". Last but not least, just mentioned flexibility had been comprehensively described in section 3.3.2. Furthermore, our definition was implemented into the project giving more than satisfiable results.

The project was completed in required time span withing gorgeous performance however this topic has so many aspects that further work should be accomplished to fully exhaust it.

5.2 Future work

Renewable energy has become more important during the last years, and it seems that it will increase rapidly in a short-term, appearing several studies concerning power and grid stability, creating new technologies and applications every time.

Advanced automation systems at homes are being developed and improved, in order to provide the inhabitants with sophisticated monitoring and control over the building's functions. "Smart Home" technologies can be linked into the aggregator system developed in this project: temperature, multi-media, security, window and door operations, as well as many other functions.

Particularly, an intelligent temperature control system could be introduced into the aggregator system, to be able to optimize the energy consumption in the houses, depending on the occupancy in the rooms, weather, doors and windows and other factor that could be interesting to take into account, having a control in each of the rooms instead of the average of the whole house. In addition, the data extracted could be studied, managed and learned by the aggregator, in order to predict occupations and consumption in different periods of the day and have a better understanding of the control of the consumption and generation of power in renewable systems and reducing the energy consumption of each house.

Finally, a remote teacher interface could be implemented in order to test the different faults handling that the aggregator has, and also adding other faults control like when there is a shutdown of the communication interface or the connection with one house is lost, as well as the actuators of the heating system not responding.

Appendix A

Contributions

Project development:

Section	Responsible	Assistant
Communication&Aggregation	Dominik	Dominik
Machine Learning	Rosa	Mirza
House Controller	Ángel	Mirza
Flexibility	Mirza	Ángel

TABLE A.1: Contributions: Project development.

Writing of the report:

Section	Responsible
Introduction and Problem Statement	Ángel
Concept description	Ángel
Methods and solution: Communication&Aggregation	Dominik
Methods and solution: Large Data	Rosa
Methods and solution: House Controller and Flexibility	Mirza
Test Cases	Ángel and Mirza
Conclusion	Dominik
Future work	Ángel

TABLE A.2: Contributions: Writing report.

Appendix B

Figures

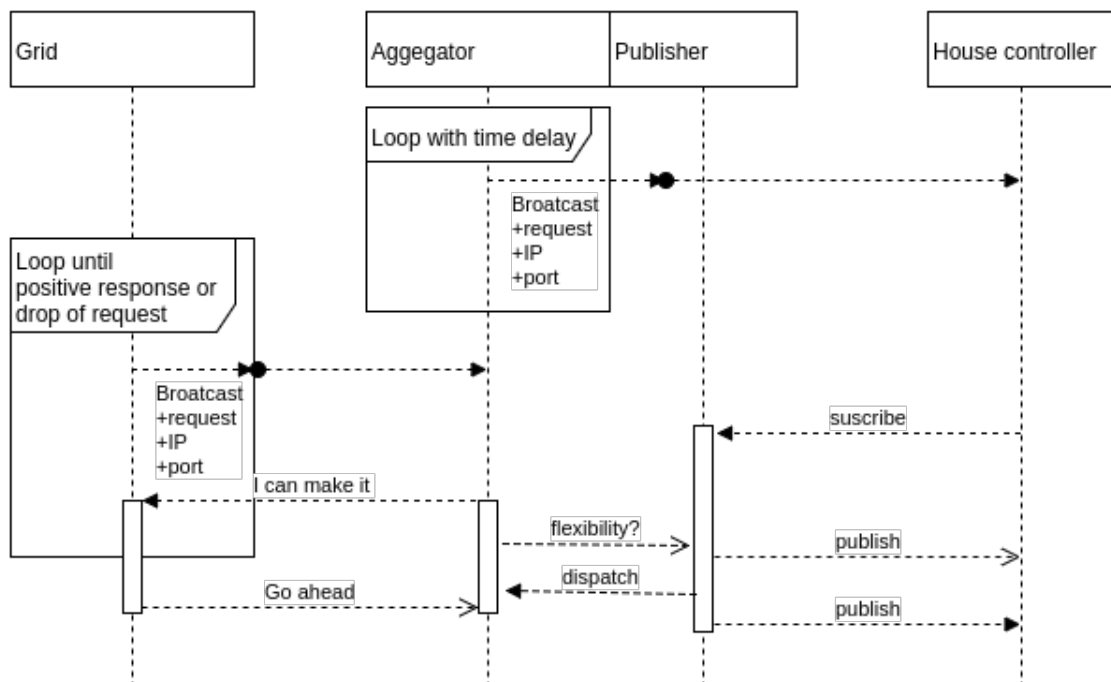


FIGURE B.1: Communication sequence in local network - dynamic solution

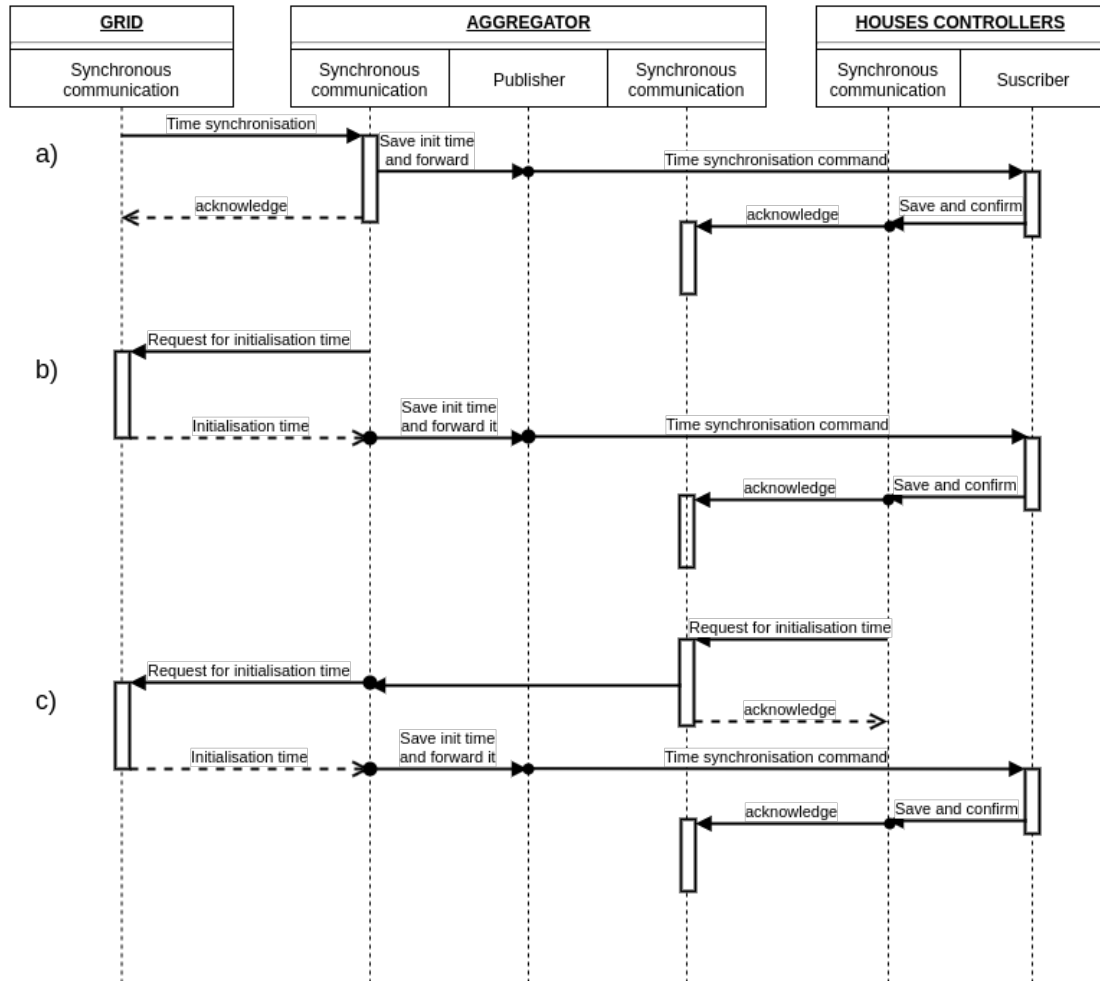


FIGURE B.2: Initial time syncing propagation invoked by a) grid b) aggregator c) house controller

```
double delta_power = 0;
double bufor_t = 0;
double bufor_p = 10000000;

Collections.sort(records.list);
System.out.println("Home name : flexibility time : flexibility power");
for (Record record : records.list) {
    if (record.flexibility_time > Double.valueOf(args[3]))
        delta_power += record.flexibility_power;
    else {
        bufor_t += record.flexibility_time;
        bufor_p = Math.min(bufor_p, record.flexibility_power);
        if (record.flexibility_time + bufor_t > Double.valueOf(args[3])) {
            delta_power += bufor_p;
            bufor_t = 0;
            bufor_p = 100000000;
        }
    }
    System.out.println(record.toString());
}
```

CODE B.1: Flexibility estimation code