Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

ON SIMULATING THE EFFECT ON THE ENERGY EFFICIENCY OF SMART GRID TECHNOLOGIES

Vincent Bakker
Albert Molderink
Maurice G.C. Bosman
Johann L. Hurink
Gerard J.M. Smit

University of Twente
Department of Electrical Engineering, Applied Mathematics and Computer Science
P.O. Box 217, 7500 AE, Enschede, THE NETHERLANDS

ABSTRACT

Most residential-used electricity is nowadays generated at inefficient central power plants consuming environmental unfriendly resources like coal or natural gas. However, a trend towards distributed generation, distributed storage and demand side load management is seen to improve the energy efficiency. In order to analyze the impact and requirements of these emerging technologies and control methodologies, good simulation models and software is required. In this paper, an improved simulator is presented to model (domestic) energy usage to analyze control strategies and improved technology on the system as a whole. Compared to the previous model, this model is more expressive and allows more future scenarios to be analyzed. Due to the added complexity, the model is extended such that the simulation can be distributed over multiple computers to reduce simulation time.

1 INTRODUCTION

Increasing the overall energy efficiency of the electricity network is an important topic in order to reduce the CO_2 footprint. Nowadays, most residential-used electricity is generated at inefficient central power plants consuming environmental unfriendly and scarce natural resources like coal or natural gas. The low efficiency is mainly caused by wasting the heat produced as byproduct and by high fluctuations in demand (de Jong et al. 2006). Since natural resources are limited and the demand for energy is still increasing, a more efficient and sustainable way to provide our energy demand is required.

As a result, a lot of research and development is taking place to improve the efficiency of electricity systems. More and more energy efficiency increasing devices and control methodologies are developed. On the generation side, a shift towards distributed generation, distributed storage and smart grids with smart control algorithms is expected. Especially technologies based on renewable sources like large windturbine- and photovoltaic (PV) parks are very promising since they provide a sustainable energy supply. On top of that, if not all energy can be supplied by such renewable sources, natural resources like oil, coal and gas have to be used as efficiently as possible. An example of an efficient domestic technology to use these natural resources is a micro Combined Heat and Power (microCHP) appliance (United States Department of Energy 2003). MicroCHP appliances consume natural gas and produce heat and electricity with a combined efficiency of around 90%.

On the demand side, controllable appliances (Hammerstrom et al. 2007) and more energy-efficient appliances can improve the overall efficiency by organizing controlled demand which can be supplied in an efficient way. Furthermore, domestic energy storage like heat buffers or batteries in electrical cars enable the possibility to shift loads of both production and consumption of energy.

Although each technology individually may improve the energy efficiency, a cooperation between these technologies can improve the efficiency even further. Controllers within the grid and within a house can optimize the runtime of all appliances, exploiting the optimization potential of these technologies. Optimization can be done

locally, like optimizing the runtime of a microCHP device, but cooperation between multiple houses and other distributed generation technologies can lead to a globally improved efficiency. Especially a combination of these devices and a large penetration rate can have a significant impact on the electricity grid.

In order to analyze the impact and requirements of emerging technologies and control methodologies, good models and simulation software is required. The simulation model must be capable of simulating, in detail, domestic technologies and control algorithms. The model has to be flexible to be able to incorporate future technologies without changing the whole model. Furthermore, it must be possible to simulate a large group of houses, since the ultimate goal is to analyze the impact of changes within the house (appliances and/or control algorithms) on the entire system. Furthermore, it must be possible to simulate and analyze requirements of future scenarios of smart grids, where control in the grid is distributed over multiple houses.

In (Molderink et al. 2009) an initial version of a simulator is presented. Although this simulator was quite flexible and fast, some drawbacks were observed. The initial version of the simulator only takes devices that consume/produce electricity and/or heat into account. Promising technologies like wind- and solar power cannot be incorporated in the simulation, limiting the possibility simulating promising future scenarios. Furthermore, it is assumed that electricity can be exchanged between houses, but that heat stays within a house. This excludes the possibility to model for example district heating or miniCHP appliances.

Besides the restrictions in the initial model, the evolution of the used control algorithms becomes a bottleneck for using this simulator. These new algorithms lead to improved results and more flexible control strategies, but their computational complexity increases as well. This leads to increased simulation time and memory requirements. As a result, simulations take a lot of time or are simply not executable due to the large memory requirements.

In the remaining of this paper, an improved version of the simulator is presented. Improvements in the model and the addition to distribute the simulation over multiple computers leads to a new simulator, overcoming the above mentioned problems. First, in Section 2 the requirements of the simulator are described. After giving some related work in Section 3, the improvement model is given in Section 4. The distribution of the simulations over multiple computers is described in Section 5. Implementation and validation details and the corresponding results are given in Section 6 and Section 7 respectively. Finally, the last section presents the conclusions.

2 REQUIREMENTS

The goal of the simulator is to create a tool to analyze the effect of residential generation and storage technologies and optimization methodologies for a large fleet of houses. Therefore, individual houses need to be modeled in detail, i.e. on a device level. The simulated situation should be an accurate model of the actual situation. Therefore, every individual (type of) device present in houses has to be modeled and it has to be possible to import measurement data (e.g. electricity usage) into the devices. However, not only current available devices need to be modeled, but it also has to be possible to incorporate future devices and technologies. Since it is not yet known what future devices, technologies and scenarios look like, the model of the house needs to be very generic and flexible. Furthermore, almost all houses are different and have different characteristics. This is caused by behavior of the residents, the devices they own and use, the number of residents in a house etc. The modeled house should be able to represent different types of houses (family/single-person houses, big/small houses, etc.). We thus need a realistic, generic and flexible model of a house.

Within a house a controller is present. This can a be simple controller like a thermostat and a controller which decides when the boiler has to fill the heat buffer, but a more sophisticated controller which takes for example electricity prices into account is also possible.

For the simulation of a massive introduction of micro-generation, multiple houses are to be combined in a grid. The base for this simulation is the realistic, generic and flexible model of a house since each house should be individually addressed due to their individual characteristics and internal state. Multiple houses grouped together form a grid, e.g. a city with a realistic mix of houses. There are a lot of different possible scenarios (different combinations of houses, housecontrollers, etc.), thus this part of the model needs to be very flexible as well. Especially when local optimizations are coordinated on a global level, i.e. the local controllers communicate with a coordinating global controller, a lot of scenarios are possible. For example, (Scott, Vaessen, and Verheij 2008) describes a situation concerning a Virtual Power Plant (VPP). Due to the different local controllers in the houses, it may be possible that not all houses respond to the global controller because not every type of local controller reacts on the signals in the same way.

Summarizing, the base of the model is a realistic, generic and flexible model of a house on a device level. Multiple houses are combined in a grid to analyze the effect of a large group of houses. The simulator should be easily adaptable to new types of micro-generators, controllers and other supported elements. It should be easy to simulate different scenarios, combinations of different houses and present local controller and global objectives.

Comparing to the requirements of the previous version of our simulator, the following requirements are still present: a) Simulation of **realistic** settings and devices. B) Both **single house** and a grid with a **large amount of houses**. C) **Flexible**, both in adding new elements and in the supported scenarios and D) **Adjustable logging/precision in time interval length**. A detailed description of these requirements can be found in (Molderink et al. 2009).

Compared to the previous simulator, it is not sufficient to concern only heat and electricity. Although the previous version already is generic and flexible, it is not possible to realistically simulate renewable sources and take the phase shift into account. Thus, the model needs to be more generic. Furthermore, improved and more complex optimization algorithms decrease the simulation speed, so also the **simulation speed** needs to be **improved**. Finally, the network communication of the optimization methodologies needs to be monitored (between local and global controller) to verify the used protocols and define the requirements for the communication media.

3 RELATED WORK

Simulation solutions already exist in a lot of different areas, e.g. optimizations for logistics, 3D modeling or process management. However, most simulation software is domain specific and is not easily ported to different application areas.

The focus of our research is on simulating the effect of (domestic) energy streams on the system as a whole. Commercial software is available where heat- and electricity load of large buildings can be simulated. These systems are often used to optimize Heating Ventilation Air Conditioning (HVAC) systems and facade control systems or to build more energy efficient buildings and take into account the structure of the building and the materials used (Pan, Zuo, and Wu 2009, Augenbroe and Hensen 2004). However, our goal is not to simulate the expected heat and electricity load, but to simulate the control methodologies to supply these loads resulting from the structure of the building. Other energy simulation software focuses on a specific technology, for example wind parks or solar cells.

We have chosen not to use simulation frameworks like Tortuga (Weatherly and Page 2004) or SimPy (Simulation in Python). Although these frameworks may provide us some generic functionality required for our simulation, they still require to create our own model within the limitations of the framework. We considered it easier and more promising to create an own model in a familiar environment and still have the flexibility to reuse work in the literature. Furthermore, our aim is to have a simulator that is very fast and memory efficient. Preferable it is usable on multiple platforms, which makes distributing the simulations easier. For this reason, we chose C++ as programming language, using Nokia's QT library. This library provides efficient, cross platform libraries for data storage, network communication and user interfaces. Furthermore, the C++ programming language enables us to use other C and C++ libraries.

In related work regarding energy-optimizing control strategies the focus is mostly on agent based approaches (Houwing and Bouwmans 2006, Kok, Warmer, and Kamphuis 2005). An example of an agent-based system is the PowerMatcher (Kok, Warmer, and Kamphuis 2005), which creates a virtual market to determine who can produce/consume energy and for which price. Important in such an approach is the stability and reliability of the bidding system, which is often simulated. The simulator described in this paper has a more generic approach, where the focus is on control of the system and the influence of that control system on the whole system. Due to the flexible design, the PowerMatcher bidding system could be embedded in our simulator.

In (Handschin and Uphaus 2005) a custom simulation system for the coordination of decentralized energy conversion is described. Like in our approach, a custom made simulator is developed. However, little details about the underlying design are given and only a device specific example is given. Our approach has a more flexible design. Different control strategies can easily be added and the design is more flexible to future technologies.

4 MODEL

This section starts with a short summary of the shortcomings of the former model for the simulation given in (Molderink et al. 2009). Next, the new model is described. We show how the shortcomings of the old model are solved and how the new model can be used in a simulation.

The model used for the simulator in (Molderink et al. 2009) is based on electricity and heat only. The goal of the new model is to improve the coverage of the model to be able to model current scenarios and future developments.

The most important limitations of the previous model are that the import of gas is disregarded (the amount and the costs), that it is impossible to model district heating, that it disregards sun and wind energy imports and that it does not take phase shifting of electricity into account. Phase shifting is not only important for islanded houses and islanded neighborhoods, but large phase shift compensations using phase-shifting loads can also decrease efficiency. Therefore, an extended and more generic model is required to cope with the limitations of the previous model. A generic and flexible approach is especially of importance since it is not yet known how future techniques and scenarios will look like.

4.1 An Improved Energy Model for Houses

Instead of taking only heat and electricity into account the model is extended to take various types of energy-related streams into account. More precisely, the new model is based on a set of *energy-types*. All energy streams within a house and energy sources used are seen as streams of energy-types: heat, electricity and gas, but also sunlight, wind, etc. Which energy-types are taken into account can be defined per instance of the model. In the previous model the only energy-types are heat and electricity. This set now can for example be extended with gas, sun and wind, but it is also possible to add reactive power to the set to be able to model phase shifting.

Within a house energy is converted (change type), (temporarily) stored and consumed by devices, where e.g. a hot water tap is assumed to be a heat consuming device just as a television is an electricity consuming device. The previous model distinguishes three different types of devices: 1) Generators generate heat and/or electricity (including grid import/export), 2) buffers store heat or electricity and 3) consumers consume heat and/or electricity. The new model distinguishes four different types of devices within the house: 1) exchanging devices, 2) converting devices, 3) buffering devices and 4) consuming devices.

4.1.1 Devices

The basis of the model are devices and energy streams between these devices: energy of a specific type flows from device to device. Every device does something with the energy-types (exchange, convert, buffer, consume). So, a device is an entity where energy flow in and/or out and where the type of these energy streams is specified.

All devices within the house are modeled individually, since the optimization algorithms to be tested with the simulations optimize behavior of individual devices. Such behavior can for instance be the decision when to run a converting device (e.g. starting a microCHP can help reducing peaks of the electricity import). Similarly, by temporarily buffering electricity, peaks can be shaved or by decoupling the heat production and consumption using a hot water tank, shifting the runtime of a microCHP becomes possible. Furthermore, demand side load management of consuming devices builds on managing individual devices.

Exchanging devices exchange energy with the environment. Concerning a house as the largest entity that is modeled, a house exchanges energy with its environment. For most conventional houses this is only electricity that can be imported and exported and gas that can be imported, but some houses also import heat from district heating. Furthermore, also sunlight and wind are modeled as energy imports. We model exchanging devices such that they exchange only one energy-type with the environment.

Converting devices convert one or more energy-types into one or more other energy-types. In our model, the amount of energy streaming into these device is equal to the amount of energy streaming out of these devices. However, energy conversions (often) have a certain amount of loss during conversion. This is modeled as a seperate energy stream out of the device. So, a microCHP for example has a gas stream in (100%) and a heat stream (80%), an electricity stream (15%) and loss stream (5%) out.

Buffering devices can temporarily store an energy-type. These devices have an energy-type stream in and the same energy-type stream out. This separation of the stream in and stream out is necessary since these streams are not alway shared, e.g. most currently installed hot-water buffers have separate in and out flow circulations. When the in and out stream are shared, this can be enforced by the characteristics of the device. Next to the in and out stream, a separate energy-type stream out can be used for modeling loss.

Consuming devices consume one or more energy-types in a certain ratio. For most devices, the amount of energy consumed in a certain time interval (the consumption profile) is a characteristic of the device and is therefore

defined on beforehand. A *loss device* is also modeled as a consuming device. For this device is not defined how much energy it consumes, it simply consumes all loss (since loss streams are connected to this device).

4.1.2 Limitations and Options

Every device has certain limitations, e.g. the amount of energy it can import, the amount of energy it can convert, etc. Furthermore, within the limitations of the device there are often multiple options possible. Examples are the timestamp a microCHP is switched on to fill a heat buffer or the option to store energy in a battery or not, etc. Every device has a set of possible options O (microCHP on/off, buffer charge or discharge and how much) and based on the state s of the device (e.g. State of Charge of the buffer) a subset O_s of the set of options are valid options. A control algorithm needs to decide which option to choose.

The limitations concerning the exchanging devices are the amount of energy which can flow in and out. They are characteristics of the exchanging devices and are modeled within the devices. The decision is how much energy is exchanged, within the bound of these limitations. The amount of every energy-type flowing in and out, as well as the limits, can be defined in kW.

The amount of energy converted by converting devices is often limited. Furthermore, often there are only a few possible levels of conversion (e.g. 0kW, 30kW and 70kW) or conversion regions (e.g. 0-20kW and 65-90kW). The efficiency can differ per level or region. The decision in this case is to choose a valid conversion level out of the set of options. Next, there are fixed ratios between the different input streams (optionally different per level). To calculate the amount of every energy-type flowing out based on the amount of energy flowing in, conversion matrices are used (again optionally different per level). Furthermore, the conversion matrices can depend on the status of the device. For example, a microCHP consumes electricity during startup and produces electricity when it is running at full speed (so it is possible to have energy streams of the same energy-type in and out).

For buffering devices the maximum amount of energy which can flow to or from the buffer depends on the State of Charge (SoC) of the buffer and the buffer size. The loss can also depend on the SoC, so a buffering device needs to keep track of its status too. The decision is how much energy flows in or out the buffer.

The amount of energy consumed by consuming devices is normally fixed during its running. However, when the energy streams are optimized the load of (some) consuming devices can be shifted in time, so it can be decided to switch off a device temporarily.

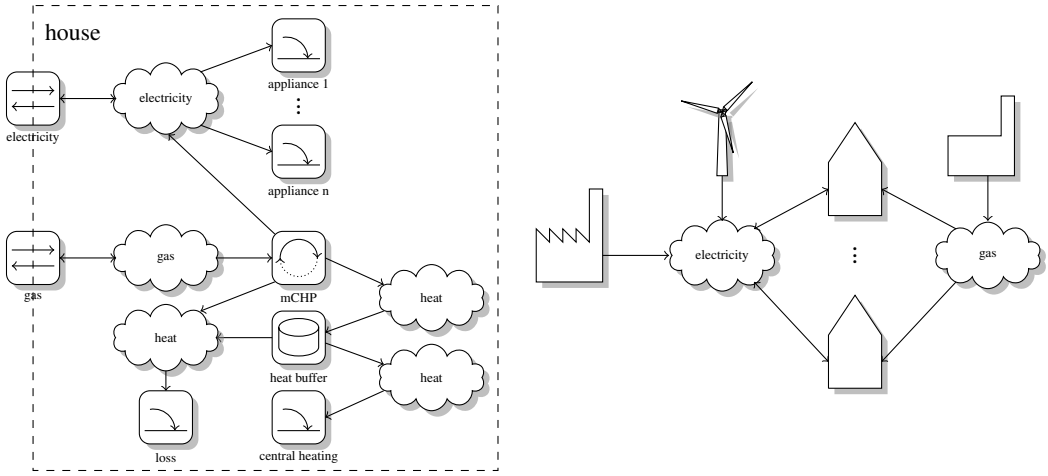
4.1.3 Streams Between Devices

As mentioned before, the basis of the model are devices and energy streams between these devices. Every device has certain energy-type streams in and/or certain energy-type streams out. Each stream consists of a flow of one energy-type. Input and output streams of devices are coupled, so energy-types can flow from one device to another. To manage these flows in a proper way, all devices are interconnected through *pools*. A pool can only transport one energy-type and has no loss. This means that, a pool has to be balanced: the amount of energy flowing in is equal to the amount of energy flowing out.

Every stream of every device is coupled to one pool and each stream in a pool must have the same energy-type. For example, in most houses all electricity producing and consuming devices are connected to one grid in the house. Electricity can flow from every electricity output stream to every electricity input stream. On the other hand, hot water flows from the boiler via a pipe to the hot-water buffer and via another pipe to the consuming devices. Here two “hot water pools” are present in the house, as depicted in Figure 1(a) (and a third one for the loss). Summarizing, within every pool energy-type is transported and every stream is connected to one pool. This introduces a lot of expression power. For example, we can model the situation that (a part of) the house is protected with an Uninteruptable Power Supply (UPS) system. The amount of energy flowing from and to a pool can be limited due to limits in the transportation medium.

4.1.4 Complete Model

The complete model of a house combines the four different types of devices and the pools. This enables the possibility to model all energy streams of a complete house. An example of such a model is shown in Figure 1(a). This house consists of multiple electricity consuming appliances, central heating, a heat buffer and a microCHP. This model of the house can easily be extended with e.g. the water stream, the reactive power or PV on the roof.



(a) Model of domestic energy streams.

(b) Model of energy streams in the network(s).

Figure 1: Improved models.

The model has to ensure that the constraints are respected. First of all, the streams should be within the bound of the limitations. Furthermore, the energy flowing in and out a device has to be in balance. The sum of the flows in and out of a converting device has to be zero and the sum of the flows in, the flows out and the change of SoC of a buffering device has to be zero. In other words, for every device a valid option $\in O_s$ has to be chosen. Finally, the energy streams within a pool have to be in balance, i.e. the sum of flows in and out should be zero. When all the pools and appliances are in balance, the complete energy balance of the house is correct.

The model so far models the devices and energy streams within a house on a detailed level and in a generic way. This is represented by the eight classes to the right in the class diagram in Figure 2 (including the controller that decides for every device which option $\in O_s$ is chosen). Next, multiple houses can be combined into a grid, similar to the combination of multiple elements into a house. The houses may exchange energy-type streams with the global pools thereby acting as active consumers. This energy needs to be delivered by energy-suppliers. It is possible that there are multiple energy-suppliers per global pool, for example a windmill (with a fixed, fluctuating amount of production) and a conventional power plant. This is shown in Figure 1(b). Such a grid model makes it e.g. possible to analyze (the impact of optimizations algorithms on) the production pattern of a power plant. Also global optimization algorithms communicating with the local optimization algorithms are modeled (see Figure 2).

4.1.5 Simulation

The presented model forms the base for simulations of the energy streams within houses over a certain period. We decided to use a discrete simulation instead of continuous simulation. A continuous simulation also requires a continuous analysis and control which is much more complex than taking a decision for a certain amount of time periods. Furthermore, continuous predictions of energy demand and production used within the optimization methods are not useful since the mismatch between prediction and real behaviour can have more impact then when discrete periods are used. Therefore, the actual controller installed in houses also works on a discrete base.

Eventbased simulation is not applicable since 1) the period to decide for is unknown (and therefore e.g. the maximum discharge current of a battery) and 2) events are highly correlated resulting in a lot of events for which the whole system needs to be analyzed (e.g. a high discharge current of a battery changes the status of the battery quickly resulting in an event). When there are a lot of such events, the advantage of eventbased simulation over discrete simulation has almost disappeared.

For a discrete simulation the simulation horizon is discretisized resulting in a set of consecutive time intervals. The number of intervals depends on the length of the planning horizon and the chosen length of the time intervals, e.g. a five minute timebase and a 24 hour horizon results in 288 time intervals. The model describes the status of the house on one moment in time, gives the limits for the following time interval and decides the amount of energy flowing in the following time interval.

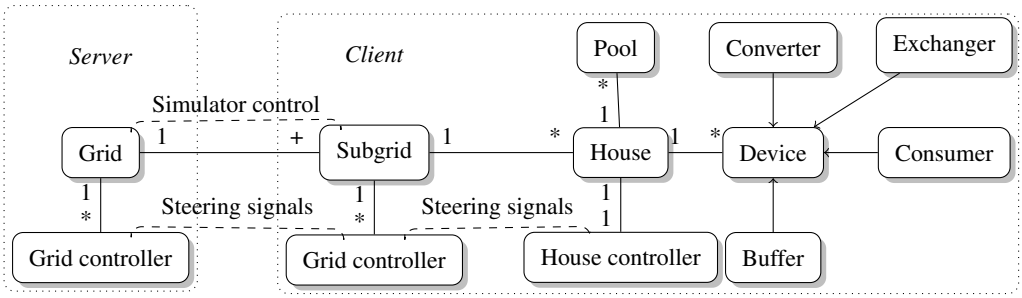


Figure 2: Design of the model and the network architecture.

In every time interval the model has to choose a valid option for the devices and the pools have to be in balance. In a normal house without intelligent control as described in Figure 1(a), the heat consumption just appears, i.e. the demand is not known in advance to the controller (e.g. a resident takes a shower). The heat demand is supplied by the heat buffer. A controller decides when to start the microCHP, i.e. when the heat buffer level drops below a certain level the microCHP is started. In a normal house the electricity consumption also just appears and together with the runs of the microCHP this gives a total electricity flow in the house. Using the current electricity infrastructure, the electricity surplus or shortage is simply exchanged with the grid. In other words, the only decision to be taken is when to start the microCHP and that decision is based on the level of the heat buffer. To simulate this basic situation, a basic *control algorithm* is added to the simulator. This control algorithm is during normal circumstances, as described in Figure 1(a), only reactive and straightforward. More sophisticated control algorithms also optimize runtimes of converting and consuming appliances and make smart use of the buffers. The overall import/export pattern of energy-types of a fleet of houses can be analyzed by combining a lot of houses into a grid where each house exchanges energy-type streams with the global pools. To reach a certain global objective with a fleet of houses, e.g. a certain overall consumption profile, the local controller communicates with a global controller, optionally in a hierarchical way.

5 DISTRIBUTED EXECUTION OF SIMULATOR

The new model is far more flexible and versatile, but it also comes with a cost of a higher complexity. The computational intensity of a single simulation has increased significantly. To keep the simulation time within reasonable limits, the computation has to be split in smaller parts and distributed over multiple computers via a network. Looking at the entities in the model, the model has a lot of oppertunities for parallelization. Each house is independent of one another, allowing these entities to be simulated on different machines. Furthermore, within a house, a house controller has to decide which options are chosen for each device. Once this decision is made, devices may need to update their internal state. These updates can all be performed in parallel.

For this reason, the simulation model is extended with a server-client model (see Figure 2). The server has three tasks. The first task is the configuration of a simulation. A configuration consists of a grid and a possible grid controller. A grid consists of a group of (different kinds of) houses. A house consists of multiple devices, where each device is connected to one or multiple pools. As mentioned earlier, each house has a house controller.

Each entity, e.g. the grid, house controllers, devices etc., can be configured within the GUI at the server. Once a configuration is built, it can be simulated. When a simulation is started at the server, it automatically distributes the houses configured in the grid into so called 'subgrids' and sends the configuration for each subgrid to each connected client.

During simulation, information is exchanged between the server and the clients. The server is responsible for keeping clients in sync, since some clients may be faster than others.

When all clients are finished, the server is responsible for aggregating all local simulation results into a global simulation result. When all data is aggregated, it can be analyzed on the server. Since the amount of data can be quite substantial, simulation results are kept at the clients as much as possible. Only the relevant information required to aggregate (sub)grids and information about the simulation results are requested by the server. For example, all houses, controllers and devices can generate simulation results. On the client, these results can be displayed in the GUI via text or plots. The GUI needs to know which information can be displayed and on which client this

information is available. When certain data is requested, the server looks up the origin of the information and sends a requests for the information to the client. This minimizes network traffic and the amount of required memory at the server. When a simulation has to be saved (to disk) for later analysis, all received information is stored directly to disk, minimizing the amount of data stored in memory.

The division of the grid into multiple subgrids enables the possibility to create an hierarchical structure. For example, multiple neighborhoods can be simulated. Each neighborhood consists of a mixture of different houses with different usage profiles. This mixture of neighborhoods can be divided into different subgrids, which can be simulated at different clients. The grid controller at the server cooperates with the grid controller of each subgrid on the clients. Each grid controller of a subgrid cooperates with both the main grid controller and each house controller. An example of such an hierarchical approach is the the creation of a virtual power plant. The goal here is to optimally control a large group of micro-generators, like microCHP appliances, to generate a certain amount of electricity. Objectives can be to minimize purchase costs of energy retailers or to ensure stability on the electricity network. Due to the large size of the fleet, it is impossible to optimally control the fleet centrally (Bosman et al. 2010). By dividing the whole control system in smaller subsystems, the subgrid controllers and house controllers can optimize the runtime of the micro-generators within the subgrid. The approach leads to a faster, more scalable system.

The added network stack provides an interface to facilitate the communication between the controllers. Another advantage of this approach is that also the communication requirements between the controllers in a real life setting can be analyzed. For example, characteristics of a certain communication medium like GPRS can be emulated. All data sent between the controllers can be sent via this emulated communication channel. Communication properties like delay, packet loss or limited bandwidth can be simulated. Using such an approach, the amount of required bandwidth or the fault tolerance of the controllers can be determined.

6 IMPLEMENTATION AND VALIDATION

As mentioned earlier, the simulator is built in the C++ programming language. A simulation consists of a grid, in which multiple (types of) houses are present. Each house has a set of devices and pools (see Figure 2). For each model-entity in the figure, a separate class is built. Flexibility is obtained by defining the devices and controllers as abstract classes. In the abstract classes minimum required functionality is implemented such that all requirements of the model are met. An implementation of an actual element consists of a class that extends the abstract class and implements the abstract defined functionality. Optionally, standard behavior can be adapted by overriding the corresponding functions. In this way the class implements the specific behavior of the element.

Although all devices have the same interface so that they can be treated equally by the controller, still four pre-defined subclasses of devices are specified. All buffers for example have a certain State of Charge, while all consuming devices have a certain demand profile determined by human behavior. All device-category specific basic behavior is implemented in the abstract classes representing Converters, Exchangers, Consumers and Buffers. Using such an approach, only limited amount of code has to be written to add a new device type to the model. To add a new device, only roughly 100 lines of codes have to be written to add a new device implementation.

Up to now, a High Efficiency Boiler, a WhisperGen microCHP appliance and a GenericConverter class are available converters. The first two classes represent real life-appliances with their device specific characteristics. For example, a WhisperGen microCHP appliance has limitations on the runtime like minimum runtime and minimum cool-down periods. The GenericConverter class is a very basic implementation which can convert one energy stream into another with a configurable efficiency, which can be used for converters without state.

The group of available consuming devices are generic appliances and a freezer. A freezer keeps tracks of the internal temperature and based on this temperature the electricity demand is determined. A generic appliance can model appliances which have a certain predefined demand profile. For example, a lamp has a fixed pattern once it is switched on. Generic appliances can be configured to have a fixed demand profile, consuming one or multiple energy-types. Their actual demand is determined by the demand profile and the moment in time at which these devices are switched on. On the start time and runtime of these appliances a stochastic variation can be added to model more realistic user profiles.

A Gledhill heat store and an implementation of the Kinetic Battery Model (KiBaM) (Manwell and McGowan 1993) are available buffer devices. The heat store can be configured to be of different sizes. Using this parameter, the effect of the size of the heat store can be analyzed. The KiBaM models a realistic battery, emulating the characteristics of a real battery.

Only a single exchanging device type is implemented. Each exchanging device can exchange one energy carrier. For each time interval of the simulation, a limitation on the amount of energy that can be exchanged can be configured. For example, a electricity connection to a house can be limited to 35A or the amount of wind imported to the house can be set to a predefined pattern.

6.1 Simulation Workflow

As mentioned earlier, a simulation can be configured. Each instance of a class within the model is responsible for generating a GUI element so that device specific parameters can configured. In the GUI, a grid can be constructed by defining which houses and how many of each house type are present in the grid. For each house it can be configured which devices and pools are present and the connection between devices and pools can be configured. Furthermore, house specific attributes like the usage of consuming devices can be configured per house.

Once a grid is configured, it can be simulated by dividing the grid into subgrids and sending the subgrids to each client. Once each client is configured, the simulation is started and the first time interval is simulated. During the simulation of a time interval, each controller performs its specific tasks and each model entity is requested to update its state. The completion of each time interval of all clients is reported back to the server to keep all clients in sync. After the last time interval, all information required at the server in order to aggregate the subgrids results and to be able to display the simulation results is sent to the server. Here, the user can interact with the GUI to display or save the simulation results.

6.2 Protocol

One of the requirements of the improved simulator was the ability to simulate larger groups of houses. By distributing the houses over multiple computers, the memory and computational requirements of the simulation are spread over multiple computers. To make the approach scalable the network overhead has to be limited, requiring a fast and efficient protocol.

Since a simulation is distributed over multiple computers, all information required for a simulation has to be available at each client. As a consequence, the protocol has to thus be able to distribute all required information prior to the construction of the (sub)grids, houses etc.

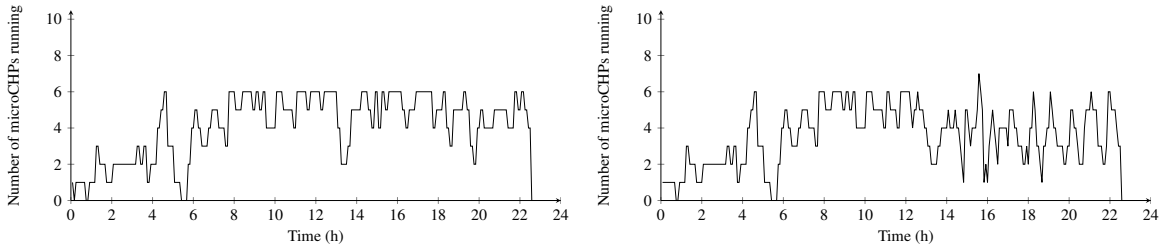
Based on the flow of a simulation, a protocol between the server and the client is developed. After connecting, a welcome message is sent to confirm that a simulation client is connected and not some random process.

When the user selects a simulation configuration to be simulated, the server automatically distributes the required houses over the connected clients. Simulation configurations are simple files, which can be easily transferred. For each client, a new configuration file of the subgrid is created, and information about this file is sent via a SimulationMessage. When the client retrieves this SimulationMessage, it first looks if it has a cached version of the configuration file. If it has a cached version, it has to check whether this file has been changed since it was obtained. This is done by sending a ConfigInfoMessage, which contains the filename of the configuration and a hash of its contents. When the server retrieves a ConfigInfoMessage, it constructs a ConfigInfoMessage from it's own configuration and sends it back to the client. When the hashes are equal, the file is up-to-date.

If the hashes are not equal, or when the configuration file did not exist at the client, a RequestConfigMessage is sent. When the server receives such a message, the file is read from disk, compressed and sent to the client. Once the configuration file is up-to-date at the client, it is scanned for dependencies. For example, a grid consists of houses and houses have their own configuration (files). For each dependency, their configuration files are exchanged in a similar way. When all configuration files are up-to-date, the client confirms it has finished the configuration phase by sending a SimulationMessage to the server.

When all clients are configured, the server sends a TickMessage to simulate the first tick. Each client simulates the time interval and when it is done it confirms replying with a TickMessage. The server waits for all clients to confirm their tick, and then sends a new TickMessage to each client. This way, all the clients stay in sync with each other, which is required when simulating a global optimization algorithm.

When all time intervals are ticked, a StoreResultsMessage is sent. During this phase, statistics and information about the whole simulation are calculated at the clients. The completion of this phase is confirmed by each client with a ResultsInfoMessage. This ResultsInfoMessage contains information about which simulation results are available at the client. Note that all the results are still stored distributed over the clients. The server only collects where the information is stored.



(a) Planning of microCHPs based on predictions.

(b) Realisation of microCHPs running based on real heat demand.

Figure 3: Results simulation interaction local and global controller.

Since the grid is split up into subgrids, the server combines the subgrids into a global grid by aggregating the data. When data of a house or grid is required, the server requests the data by sending a RequestDataMessage to the client which has this data. The client sends the required information to the server with a DataValueMessage. After aggregating all subgrid data into a global grid, the GUI on the server is informed that the simulation has completed.

When the simulation has to be saved to disk, or when information about the simulation has to be displayed in the GUI, the server requests the required data from the clients on demand. This way, the amount of memory required on the server and the amount of data transferred over the network is limited.

By disconnecting the client, the simulation results are discarded at the client. Since it might be possible another simulation has to be executed, the client automatically reconnects to the server, and the process starts again from the beginning. Using this approach, a client can be installed on many machines, creating a large cluster of simulation clients. Since the discovery of the server is completely automatic, no user interaction at the client is required.

6.3 Validation

As mentioned in previous sections, the amount of energy that flows during a time interval is determined by the energy demand, the amount of available (stored) energy and the production capacity of the producers. The house controllers are responsible for selecting the right set of options during simulations. As described in (Molderink et al. 2010), the model can be expressed using a mixed integer problem (MIP). Using this approach, the validity of the model can be ensured by adding the proper constraints to the MIP. For example, the balance within the pools can be easily expressed by requiring the sum of the energy flow in the pool to be zero.

The correctness of the implementation of the other classes is checked using unit testing.

7 RESULTS

To test the simulation, two different simulations are executed. The first one is a simulation of ten houses using global and local controller with as global objective to prevent too high production. Such a scenario was the goal of the previous simulator. When this scenario can be simulated the simulation capabilities of the previous simulator are covered. The second simulation uses the same houses but different grid sizes to tests the simulation speed.

7.1 Interaction Local/Global Controller

The ten houses used for this simulation are the houses shown in Figure 1(a). Each house consists of multiple electricity consuming appliances, a heat consuming appliance (central heating), a heat buffer and a microCHP. The objective is to act as a Virtual Power Plant: a on beforehand determined planning of the runtimes of the microCHPs should be met. The planning is based on prediction of the heat usage. A more detailed description of the scenario and the local and global optimization algorithms can be found in (Molderink et al. 2009). The results are shown in Figure 3. First of all, it is possible to run this simulation so the capabilities of the previous simulator are covered. Furthermore, the results of the schedule are much better than with the previous simulator, there is less deviation from the planning. This is caused by some optimizations in the local controller (especially improved parameters for the cost functions). All deviation from the planning is now only caused by prediction errors.

Table 1: Simulation speed (seconds).

	1 house	100 houses	500 houses	1000 houses
standalone	1	47	234	490
network, one client	1	50	246	491
network, two clients	-	28	123	245
network, three clients	-	20	88	170

7.2 Simulation Speed

To simulate a large fleet of houses a lot of computational power is required. This is especially caused by the local control algorithm that is executed every time interval for every house. To measure the speed and the influence of the network layer the above described scenario is simulated for 1, 100, 500 and 1000 houses. It is simulated using the standalone version, the network version with one client (measuring network overhead) and the network version with multiple clients with similar specifications to analyze the speedup. Due to some memory optimizations in the code, the memory usage is no limitation anymore. The memory usage stays within a couple of hundreds of megabytes.

The simulation times are shown in Table 1. It can be seen that the network overhead is negligible for large simulation instances and the speedup using the network version is significant. Simulating larger instances leads to less network overhead, since the aggregation of the subgrids is determined by the amount of clients, not the amount of houses. Due to the synchronization in the protocol, the speedup is limited to the slowest client.

8 CONCLUSIONS

Results show that the same scenario as used in (Molderink et al. 2009) can be simulated in the new model and that the performance of the evolved house controller yields to better results. Furthermore, the extended model presented in this paper shows enough expression power to simulate more future energy improving technologies. Especially the general and flexible interface to create a control strategy gives room for good analysis of the impact of different control algorithms. The requirements set in Section 2 are met, although the simulation time of the simulator still has room for improvement. The added network layer results in a speed-up almost similar to the amount of clients used, yielding to acceptable simulation times. Furthermore, the added network layer allows the analysis of network requirements of different control strategies.

ACKNOWLEDGEMENTS

This research is conducted within the Islanded House project supported by E.ON UK and the SFEER project (07937) supported by STW, Essent and Gasterra.

REFERENCES

Augenbroe, G., and J. Hensen. 2004. Simulation for better building design. *Building and Environment* 39 (8): 875 – 877. Building Simulation for Better Building Design.

Bosman, M., V. Bakker, A. Molderink, J. Hurink, and G. Smit. 2010. On the microCHP scheduling problem. In *Third Global Conference on Power Control and Optimization*: IEEE.

de Jong, A., E.-J. Bakker, J. Dam, and H. van Wolferen. 2006, Juli. Technisch energie- en CO2-besparingspotentieel in Nederland (2010-2030). *Platform Nieuw Gas*:45.

Hammerstrom, D., R. Ambrosio, T. Carlon, J. DeSteese, G. Horst, and R. Kajfasz. 2007, July. Pacific northwest gridwise testbed demonstration projects, part i and ii. Pacific Northwest National Laboratory.

Handschin, E., and F. Uphaus. 2005, June. Simulation system for the coordination of decentralized energy conversion plants on basis of a distributed data base system. In *Power Tech, 2005 IEEE Russia*, 1–6.

Houwing, M., and I. Bouwmans. 2006, December 4-8. Agent-based modelling of residential energy generation with micro-CHP. In *2nd International Conference on Integration of Renewable and Distributed Energy Resources, Napa, CA, USA*.

Kok, J., C. Warmer, and I. Kamphuis. 2005, Jul. Powermatcher: Multiagent control in the electricity infrastructure. In *4th international joint conference on Autonomous agents and multiagent systems*, 75–82: ACM.

Manwell, J., and J. McGowan. 1993. Lead acid battery storage model for hybrid energy systems. *Solar Energy* 50 (5): 399–405.

Molderink, A., V. Bakker, M. Bosman, J. Hurink, and G. Smit. 2010. On the effects of MPC on a domestic energy efficiency optimization methodology. Submitted to IEEE international Energy Conference 2010, Kingdom of Bahrein.

Molderink, A., M. Bosman, V. Bakker, J. Hurink, and G. Smit. 2009. Simulating the effect on the energy efficiency of smart grid technologies. In *Proceedings of the 2009 Winter Simulation Conference*: IEEE.

Pan, Y., M. Zuo, and G. Wu. 2009, July. Whole building energy simulation and energy saving potential analysis of a large public building. In *Eleventh International IBSPA conference*, 129–136.

Scott, J., P. Vaessen, and F. Verheij. 2008, Apr. Reflections on smart grids for the future. Dutch Ministry of Economic Affairs.

United States Department of Energy 2003, December. The micro-CHP technologies roadmap. *Results of the Micro-CHP Technologies Roadmap Workshop*.

Weatherly, R., and E. Page. 2004, Dec. Efficient process interaction simulation in Java: implementing co-routines within a single Java thread. In *Proceedings of the 2004 Winter Simulation Conference*, Volume 2, 1437–1443.

AUTHOR BIOGRAPHIES

VINCENT BAKKER received his M.Sc. degree in Computer Science from the University of Twente in 2007, with a minor certificate in Entrepreneurship. Currently he is working on his Ph.D. thesis researching domestic demand prediction for in home optimizations. Currently his interest are: machine learning, optimization modeling and large scale distributed (intelligent) systems. His email is <v.bakker@utwente.nl>.

ALBERT MOLDERINK received his B.Sc. and M.Sc. degree in Computer Science from the University of Twente in respectively 2004 and 2007. After completing his study he started working towards a Ph.D. degree at the University of Twente under supervision of Prof. dr. ir. G.J.M. Smit of the Computer Architecture of Embedded Systems (CAES) group, faculty of Electrical Engineering, Mathematics and Computer Science. His research interests include energy efficiency, mathematical modeling and optimizations, algorithm development and embedded hardware.

MAURICE G.C. BOSMAN received his M.Sc. degree in Applied Mathematics from the University of Twente in February 2008. Currently he is a PhD student in the CAES and DMMP groups at the faculty of Electrical Engineering, Mathematics and Computer Science at the University of Twente. His research interests include energy efficiency, scheduling and online algorithms.

JOHANN L. HURINK received a Ph.D. degree at the University of Osnabrueck (Germany) in 1992 for a thesis on a scheduling problem occurring in the area of public transport. From 1992 until 1998 he has been an assistant professor at the same university working on local search methods and complex scheduling problems. From 1998 until 2005 he has been an assistant professor and from 2005 until 2009 an associated professor in the group Discrete Mathematics and Mathematical Programming at the department of Applied Mathematics at the University of Twente. Since 2009 he is a full professor of the same group. Current work includes the application of optimization techniques and scheduling models to problems from logistics, health care, and telecommunication.

GERARD J.M. SMIT received his M.Sc. degree in electrical engineering from the University of Twente. He then worked for four years in the research and development laboratory of Océ in Venlo. He finished his Ph.D. thesis entitled "the design of Central Switch communication systems for Multimedia Applications" in 1994. He has been a visiting researcher at the Computer Lab of the Cambridge University in 1994, and a visiting researcher at Lucent Technologies Bell Labs Innovations, New Jersey in 1998. Since 1999 he works in the Chameleon project, which investigates new hardware and software architectures for battery-powered hand-held computers. Currently his interests are: low-power communication, wireless multimedia communication, and reconfigurable architectures for energy reduction. Since 2006 he is full professor in the CAES chair (Computer Architectures for Embedded Systems) at the faculty EEMCS of the University of Twente.