Fog Paradigm for Local Energy Management Systems

Amir Javed[1], Omer Rana[1], Charalampos Marmaras[2], and Liana Cipcigan[2]

[1] School of Computer Science & Informatics, Cardiff University, UK
javeda7@cardiff.ac.uk
[2] School of Engineering, Cardiff University, UK

Abstract. Cloud Computing infrastructures have been extensively deployed to support energy computation within built environments. This has ranged from predicting potential energy demand for a building (or a group of buildings), undertaking heat profile/energy distribution simulations, to understanding the impact of climate and weather on building operation. Cloud computing usage in these scenarios have benefited from resource elasticity, where the number and types of resources can change based on the complexity of the simulation being considered. Many such energy simulations require resource scalability and on-demand resources to meet the computational and data storage requirements. While there are numerous advantages of using a cloud based energy management system, there are also significant limitations. For instance, many such systems assume that the data has been pre-staged at a cloud platform prior to simulation, and do not take account of data transfer times from the building to the simulation platform.The need for supporting computation at edge resources, which can be hosted within the building itself or shared within a building complex, has become important over recent year. Additionally, network connectivity between the sensing infrastructure within a built environment and a data centre where analysis is to be carried out can be intermittent or may fail. There is therefore also a need to better understand how computation/analysis can be carried out closer to the data capture site to complement analysis that would be undertaken at the data centre. We describe how the Fog computing paradigm can be used to support some of these requirements, extending the capability of a data centre to support energy simulation within built environments.

Key words: Distributed Clouds, Fog Computing, Energy Management, Built Environments

1 Introduction

Buildings alone are considered to be one of the largest contributors of total energy consumption and towards greenhouse gas production [27],[15]in most IEA(International Energy Agency) countries [1].Having said that they are also considered to offer the greatest potential for achieving significant greenhouse gas emission reductions [26],[39]. For these reasons improving energy efficiency of buildings has received a lot of attention globally [40]. Looking at a smaller scale, improving energy use in buildings also leads to significant cost savings for the building manager, increasing the utility of the paid energy unit.

According to the US National Institute of Standards and Technology (NIST), buildings are an integral part of a Cyber-Physical system (defined as a co-engineered interacting network of physical and computational components) [24]. Being part of this smart ecosystem, buildings need to integrate with future smart grids and transform their simplistic consumption-only profile to a complex one that includes local distributed energy resources (DER) and/or storage. Depending on the available assets, building managers must change from typical consumers to ”prosumers”, being able to both produce and consume energy on their premises. Nowadays building are equipped with building energy management systems (BEMSs), which control the heating, ventilation, air conditioning (HVAC) and lighting systems to reducing energy consumption while maintaining occupants comfort [40].However, due to these ever-changing and uncertain indoor and outdoor characteristics, the performance of typical BEMS often falls short of expectation, lacking the necessary data processing, evaluation, and control methodologies[2], [41].

In order to achieve this objective, there is a need to develop an efficient energy management system. An energy management system that allows a building to be part of a smart ecosystem. To be seen as an element of a miniature (local) energy system that interacts with the other elements of the system in a coordinated fashion. A typical BEMS that tries to optimise the buildings energy efficiency will always fail (in the general context), if other elements of the system are not considered. On the other hand, considering the building in a local energy system with other buildings, distributed energy resources, energy storage and EVs, the energy management and coordination becomes more difficult. Over the years researchers have proposed many cloud based energy management system to overcome challenges such as single point of failure, prone to distributed denial of service attack , limited service capability due to single server, limited memory and computational resources[13][16][43][28]. These cloud based energy management system are capable for multidimensional data analysis for smart grids[19] or do data management and parallel processing in real time [18]. Other types include storing data on cloud so it can improve the service [29]. The rationale to use cloud in each system was for provide on demand processing and storage capability to improve the services provided by energy management system.

However, as we move data processing and storage to cloud we are faced with issues like latency[5] and disruption of services due to host unreachable. One of the possible solution to overcome these challenges is by using fog computing[4] that extends cloud computing and service at network edge.Fog computing enable developers to deploy services at the network edge using devices such as set-top-boxes or access points. The rationale of using fog computing is to process data at the network edge using a distributed architecture without effecting the quality of service. In the past we have seen evidence of Fog computing being used in context of smart grids for load balancing [38] and real time processing of energy data [5].

However, it is necessary to better understand how to cope with the intermittent characteristics of the different elements of such a built environment, and handle the large amounts of data that is generated from various monitoring infrastructure associated with such an environment. Predictive models should also be incorporated in the coordination mechanism in order to deal with the associated uncertainties and increase the control efficiency. From the building managers perspective, these algorithms must be able to facilitate different control strategies according to the overall coordination objective and enhance the (self-)awareness of the system. In this paper we propose a cloud based local energy management system that is used (i) to flatten the demand profile of the building facility and reduce its peak, based on analysis that can be carried out at the building or in its vicinity (rather than at a data center); (ii) to enable the participation of the building manager in the grid balancing services market through demand side management and response. Furthermore,the Local Energy Management System (LEMS) is extended using the Fog computing paradigm for the holistic management of buildings, energy storage and EVs.

2 Related Work

Over the year an efficient energy management system has been focus of research for building or set of building using smart meters. Researchers have suggested improvement to energy management system by focusing and incorporating components such as Building energy management system, home energy management system , shifting of energy load and looking at dynamic pricing [10]. To over come challenges in conventional energy management system, such as central point of failure or scalability due to limited memory and limited bandwidth to handle large request [3], researchers have proposed cloud based energy management system. Keeping the challenges in mind and to overcome them a cloud based demand response system was proposed that introduced data centric communication and topic based communication models[13]. Their model was based on a master and slave architecture in which the smart meters and energy management system at home acted as slave where as the utility acted as masters.The authors advocated that a reliable and scalable energy management system can be built using their model. In another approach the energy management system was built by considering the energy pricing to be dynamic[16]. While building this model the authors considered the peak demand of the building and incorporated the dynamic pricing while handling customer requests. In another approach researcher had proposed an architecture for control, storage , power management and resource allocation of micro-grids [28] and to integrate cloud based application for micro-grids with external one.The bigger and distributed the smart grid infrastructure become the more difficult it is to analyse real time data from smart meters. [43] suggested a cloud based system is most appropriate to handle analysis or real time energy data from smart meters. In another approach power monitoring and early warning system facilities were provided using cloud platform [12]. A mobile agent architecture for cloud based energy management system was proposed to handle customer request more efficiently [33]. In another approach a dynamic cloud based demand response model was proposed to periodically forecast demand and by dynamically managing available resources to reduce the peak demand [31].

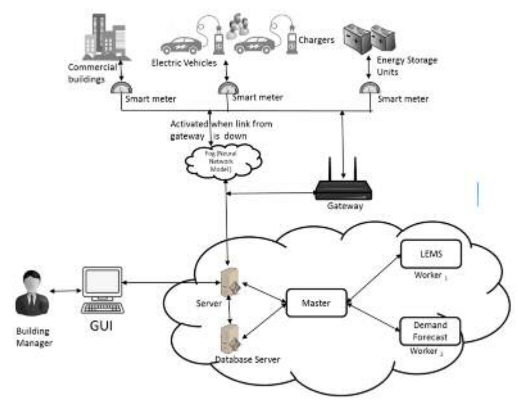
The concept of Fog computing has emerged to tackle issues relating to cloud latency, location awareness and improve quality of service for real time streamed data. Fog computing has been used in the context of smart grids in a number of ways. However, while using fog computing it is quite important to understand the energy sustainability for devices that are part of fog network. To address proper energy efficient management strategies [9] proposed a energy-aware management strategy that is able to improve the energy sustainability of entire federated IoT cloud ecosystem.Furthermore, fog computing has been applied in smart grids for enabling energy load balancing applications to run at the network edge, especially in close proximity to smart meters within micro-grids [38]. Fog-based infrastructure is used in this case to estimate energy demand, identify potential energy generation and the lowest energy tariff at that instant – using this information to switch to alternate sources of energy in a dynamic manner. Furthermore, a fog infrastructure is also used as *collectors*, to gather energy related data, and subsequently process this in real time data and generate command signals to consumer devices [5]. At the network edge the actual computational processing carried out is limited, while most of the data is pushed to a cloud data center to generate real time reports and for visualization. Fog-based systems have also been used to *control* building energy consumption. In this scenario of use, sensors installed at various places read data related to humidity , temperature and various gases in the building atmosphere. Based on these readings, the sensors will work together to reduce the temperature of the building by activating or deactivating various electrical devices [32].

Mobile Fog was proposed to tackle latency issues for geo-spatial distributed applications [11], comprising of a set of computational functions that an application executing on a device can invoke to carry out a task. However the functions supported by a *mobile Fog* infrastructure are not general purpose, but are application specific. Similarly, to tackle latency issues another proposal was presented in [25] by focusing on the placement and migration method of Fog and Cloud resources. This work describes how complex event processing systems can be used to reduce the network bandwidth of virtual machines that are migrating. In mobile Fog concept a local cloud is formed by combining capacity across neighboring nodes of a network [23] and one resource within these nodes acts as a local resource coordinator. In [23], the authors propose a framework to share resources based on the availability of particular service-oriented functions. Madsen et al. [20] combine smart grid, Cloud, actuator and sensors together for a reliable fog computing platform. To address demand side response within a smart grids, Maharjan et al. [21] worked on maximizing the benefits to both consumer and power companies using a game theoretic approach. Their work was based on a Stackleberg game [14] between the consumers and the energy companies so that both the parties were able to satisfy their utility functions. In a similar approach, a cloud based framework was presented for a group of energy hubs, to manage two way communication between energy companies and consumers to reduce peak to average ratio of total electricity – while at the same time requiring consumers to pay less by using a game theoretic approach [30]. Similarly, [8] describe an approach for reducing the energy consumption by investigating the interaction of consumers and energy companies. Based on the interaction of the consumer and power companies they proposed a game theoretic energy schedule, the aim of which was to reduced the peak to average power ratio. In another approach a user aware demand management system was proposed [42] that manages residential load while taking into account user preferences. The proposed model used both energy optimization model and game theoretical model to maximize user saving without causing any discomfort. Most of these approaches require use of batch-oriented processing, generally at a data centre. The concept of cloud based energy system has been used to over come challenges that a conventional micro-grid based energy management system faces. However, while moving to cloud is advantages it has some challenges as well and one of them being loss of communication failure between the cloud based energy management system and endpoints (electric vehicles and energy storage units). We address this issue by suggesting how fog paradigm can be used along with cloud based energy management system to overcome situation of communication failure.

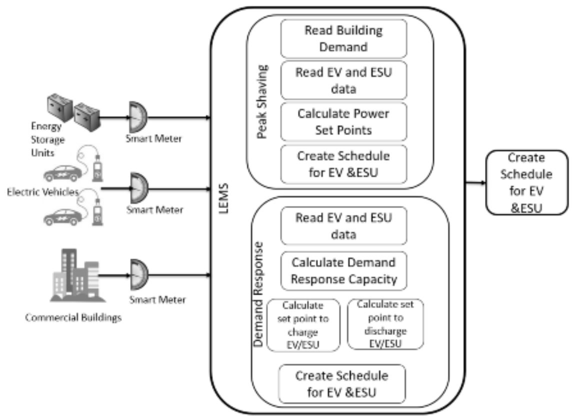
3 Cloud based Local Energy Management Systems

3.1 LEMS components

The proposed Local Energy Management System (LEMS) architecture is presented in Fig. 1.With reference to Figure 1b, the LEMS manages the demand from electric vehicles (EVs) and energy storage units (ESU's) at building premises by sending power set points through a Gateway to EV chargers. An electricity demand forecast (software) tool was developed in order to work with the LEMS, which estimates the electricity demand of a building over a particular time



(a) Architectural layout of Cloud based Local Energy Management System and its components using Fog computing


Energy Storage Units
Smart Meter
Electric Vehicles
Smart Meter
Commercial Buildings
Smart Meter
LEMS
Peak Shaving
Read Building Demand
Read EV and ESU data
Calculate Power Set Points
Create Schedule for EV &ESU
Demand Response
Read EV and ESU data
Calculate Demand Response Capacity
Calculate set point to charge EV/ESU
Calculate set point to discharge EV/ESU
Create Schedule for EV &ESU
Create Schedule for EV &ESU

(b) Architecture of Local Energy Management System

Fig. 1: Architecture of Cloud Based Energy Mangement System

period. The software tool uses historical data (collected from actual building use) and the weather measured in the proximity of the building.

The LEMS operates in timesteps during which the system is considered static, i.e. any changes are only discovered at the end of the timestep. A timestep of 15 minutes is used in this work, providing a trade-off between a dynamic (semi-real time) and a reliable operation that allows the frequent capture of the building conditions and minimizes risk of communication lags. Data about EVs located at the building, such as their battery capacity, state of charge (SoC), expected disconnection times, charging/discharging power rate, charging/discharging schedule and available discharge capacity, is requested from charging stations upon the connection of every EV. Information regarding the available capacity, state of charge (SoC), charging/discharging power rate and charging/discharging schedule is requested from every ESU. This information is stored in a database, and is accessed from the LEMS after every timestep in order to define future power set points for charging stations.

A pre-forecast analysis stage was included in order to increase the performance of the demand forecast tool, by including weather information in the forecasting process. The objective of this stage is to identify the optimal number of weather attributes to be considered by the model to improve accuracy. Using historical local weather data and building energy consumption data, an analysis was performed in order to calculate the correlation of the available weather data with the electricity demand of each building.

The forecast model used an artificial neural network, implemented using the WEKA toolkit [27]. Electricity consumption on a random day was forecast for each building for every timestep. The forecast accuracy was calculated using the mean absolute percentage error (MAPE) metric. For each building the set of attributes that resulted in the least MAPE was selected as the optimal one. The model performs a day-ahead power demand forecast using the optimal ANN configuration suggested by the pre-forecast analysis model.

LEMS has been deployed on the CometCloud [6] system. CometCloud enables federation between heterogeneous computing platforms that can support complete LEMS work, such as a local computational cluster with a public cloud (such as Amazon AWS). There are two main components in CometCloud: a *master* and (potentially multiple) *worker* node(s). In its software architecture, CometCloud comprises mainly three layers: the programming layer, a management layer and a federation or infrastructure layer. The software layer identifies tasks that needs to be executed, the set of dependencies between tasks that enables a user to define the number of resources that are available along with any constraints on the usage of those resources. Each task in this instance is associated with the types of LEMS operation supported, or whether a demand forecast needs to be carried out. In the management layer policy and objectives are specified by the user that help in allocating the resources to carry out tasks. In addition to allocation of resources this layer also keeps a track of all tasks that are generated for workers, and how many of these have been successfully executed [7]. In the federation layer a lookup system is built so that content locality is maintained and a search can be carried out using wildcards[22]. Furthermore a "CometSpace" is defined that can be accessed by all resources in the federation [17]. CometCloud uses this physically distributed, but logically shared, tuple space to share tasks between different resources that are included in a resource federation. The main task of the master node is to prepare a task that is to be executed and give information about the data required to process the task. The second component is the worker, which receives tasks from the master, executes the job and sends the results to the place specified by the master. In our framework there are two workers – one that will be running the LEMS algorithm that will generate the schedule and the second that will forecast energy demand for the next day, to generate the charging and discharging of the electric vehicles.

There are two cloud-hosted servers that receive requests from a graphical user interface, and based on the requests call the appropriate function via the master. The second server manages a database which contains information about building data, EVs and weather attributes around the building. The database is used to store historic data about power consumption, energy pricing etc. for each building and information regarding the weather is used to forecast (energy) demand for the next day. There is an intermediary gateway which intercepts all signals from the cloud server and forwards the requests to the EVs to either charge or discharge.

The last component is the Fog server that contains sensors to read data at every time step from buildings, EVs and from energy storage units. There is a pre-generated model placed inside the Fog component to which the data read from the sensor are sent, and the output is the schedule for each EV for the time step. The rationale for using the Fog component is that in case there is network failure or latency, such that a signal is not able to reach the charging station, then with the help of sensors the current state of each asset can be read and a schedule can be prepared. The predictive model is updated every 24 hours, replacing any previous estimation that was generated.

3.2 Energy Management System Operation

The LEMS maximizes its utility to the building manager by adjusting its operational target (objective) according to the system status and condition. Two scheduling algorithms for the management of the EVs and the ESUs were designed, namely Peak Shaving Schedule and Demand Response Schedule. Each algorithm serves one objective and the LEMS shifts from one scheduling strategy to another depending on the objective of the building manager:

– **Peak Shaving schedule:** This approach aims to flatten the aggregate demand profile of the building facility. This is achieved by *filling the valleys* (i.e. at periods of low demand) and *shaving the peaks* (i.e. at periods of high demand) of the demand profile using controllable loads (EVs, ESUs) of the building facility – this approach aims to shift energy load from periods where demand/tariff is likely to be high, to periods of low demand (e.g. night). Refer figure 1b The LEMS reads the building demand for next timestep, current charge in EV and ESU. Then calculates the charging / discharging schedules of the EVs and ESUs, and sends them the corresponding power set points at the beginning of every timestep. These power set points are messages with the exact power rate at which each EV/ESU must charge/discharge at each timestep.
– **Demand Response schedule:** This approach is intended to enable building managers to participate in the ancillary services market and provide demand response actions to the grid. It was assumed that the system operator sends requests to the building manager to either reduce or increase its aggregate demand in the next time step (of 15 minutes). Similar to peak shaving mode, refer figure 1b, in demand response mode LEMS read the current charge in EV's and ESU's to calculate available energy that can be used to increase or decrease demand. Then based on the request received the LEMS overrides the charging/discharging schedules of the available controllable assets by creating a charging/discharging schedule for all connected assets.

3.3 Fog Paradigm to tackle communication failure between LEMS and Charging stations

In the *Peak Shaving* mode, the highest demand of a building is flattened by creating a schedule to manage the charging/discharging of EVs and ESUs, so that the discharging operation takes place when the building demand is highest, and charging operation takes place when the demand is low. A user can enter the peak shaving mode by making a selection from a graphical user interface (as illustrated in figure 1), leading to the generation and submission of a computational task to the CometCloud system. On completion of these tasks, a peak shaving schedule is created and forwarded at the end points through the gateway. A day-ahead (24 hour) schedule is created to ensure that charging stations can make local decisions even if there is a communication failure to the Cloud-hosted LEMS. The day-ahead schedule is based on the number of connected EVs and ESUs observed at the time of creation of the schedule (and estimating the number of EVs/ESUs to be observed for the next day).

However, as the number of EVs can change during the day, and the EV/building demand can be inaccurate, the LEMS is programmed to collect data from buildings and connected EVs/ESUs at regular intervals (defined by the timestep parameter previously described). For our experiment, this timestep is set to 15 minutes but can be changed by a building manager. At each time step, new data is read and the schedule updated for the next 24 hours. This will ensure that any changes in the environment are taken into account (e.g. new EV arrival or departure) by LEMS and the charging station will always have a 24 hours schedule updated every 15 minutes.

When LEMS operates in the demand response mode, a user can request an increase or decrease in total demand using a graphical interface. The computation of this request operates similarly to the previous scenario – i.e. a computational task is generated, executed on the cloud platform, and the result (a schedule) sent to the gateway. Based on the request a schedule is prepared to based on available EVs/ESUs. However, the demand response schedule (generated by LEMS) is only valid for one time step, with the schedule reverting back to peak shaving mode after this time step. For example, if a user has made a *demand down* request then LEMS will create a schedule that will discharge all available connected EVs/ESUs for one time step. However the schedule for other remaining intervals over the remaining 24 hours will revert back to peak shaving operation. This indicates that during a communication failure (at a particular time step), the demand response request will not be executed and will have to be made again.

Currently LEMS handles communication failure by creating a 24 hour moving window of charging/discharging schedule for connected EVs/ESUs. Furthermore, this schedule gets updated every time step, so that if the number of EVs has changed in the last 15 minutes then this change is taken into account. However, LEMS does not address a situation when there is a communication failure between LEMS and the charging point, and during this communication failure a new EV arrives. As there is no communication link available, no new schedule can be created for the new EV. To address this situation we create a forecasting model that can be hosted at the edge of the network (i.e. closer to the charging points for EVs) that will, by reading the environment data (information about EV, building energy and time) available locally, predict the charging schedule over one time step, or make adjustments (using pre-defined rules) to an existing schedule generated by LEMS if there is a communication failure.

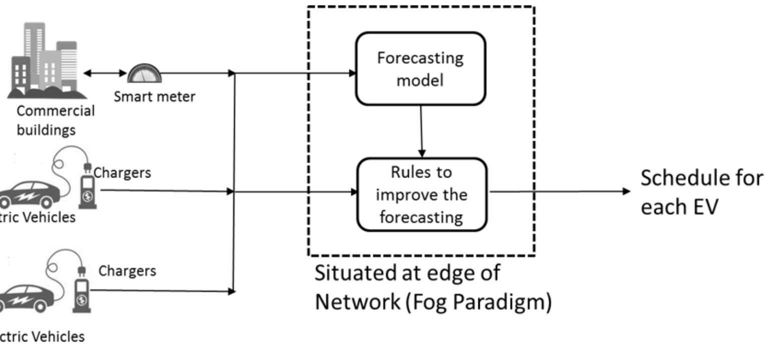


Fig. 2: Flow diagram

Figure 2 summarises how an EV charging schedule is produced in case of a communication failure, and when the number of EVs have changed based on the originally generated schedule. There are two main components that interact to update the EV charging schedule:

– the forecasting model developed using the LEMS system – using attributes such as time of day, EV identity, state of charge of EV at that time, the current energy consumption of the building, in order to estimate the charging schedule for that EV at that time. A number of learning algorithms were used to determine the schedule, and the model with the lowest error rate was placed at the edge of the network. Once the model (used to derive a schedule) is created, a set of rules that govern the charging and discharging of the vehicle are used to update the model, providing an alternative means to alter the LEMS generated schedule using local information (in case of network failure).

– The rule based component is used to improve the accuracy of the forecasting model. The rule based component takes the LEMS-generated schedule that was saved at the charger, and produces a new charging schedule as output. Once all the attributes/parameters (as indicated above) are passed as input to the algorithm, it checks if there is already a previously generated schedule for the connected EV, and updates it according to the rules available. If there is no schedule for the connected EV (using an EV identifier), then this indicates that this EV arrived after the communication failure and the LEMS-generated schedule did not take this EV into account. The first things the algorithm checks is the state of charge of the vehicle (SOC_EV_{it}) at time instant 't', if this is less than the minimum state of charge (Min_SOC), then a charge event is generated (regardless of the LEMS-generated schedule) for that time step. Furthermore, the algorithm checks if the time to departure of the vehicle is less than 30 minutes, and the forecasting model is attempting to discharge the vehicle, then the schedule is updated to generate a charging event.

Algorithm 1: Algorithm

```
Input: Scheduling algorithm generated by forecasting model for each EV denoted by $Sch\_EV_{it}$ (where i denotes the
       number of EVs and t denotes time instant t),
Schedule stored at Charger generated by LEMS $Old\_Sch\_EV_{it}$,
State of Charge of each EV $SOC\_EV_{it}$,
T: time of day,
Minimum State of Charge for each EV $Min\_SOC$,
Maximum State of Charge for each EV $Max\_SOC$,
Identifier for each connected EV $EV\_ID_i$ ,
time of (each) EV departure $Time\_EV_i$
Output: Output New Schedule for each EV as denoted by $New\_Sch\_EV_{it}$
 1: for i in tolal EV do
 2:   $charge = 3$
 3:   $Discharge = -3$
 4:   $idle = 0$
 5:   if $EV\_ID_i$ in $Old\_Sch\_EV_{it}$ then
 6:     set $New\_Sch\_EV_{it}$ = $Old\_Sch\_EV_{it}$
 7:   else if $Sch\_EV_{it}$ == Discharge and $SOC\_EV_{it} < Min\_SOC$ then
 8:     set $New\_Sch\_EV_{it} = charge$
 9:   else if ($Sch\_EV_{it}$ == Discharge or $Sch\_EV_{it}$ == idle) and $(Time\_EV_i - t) < 30$ then
10:     set $New\_Sch\_EV_{it} = charge$
11:   else if $Sch\_EV_{it}$ == charge and $SOC\_EV_{it} = Max\_SOC$ then
12:     set $New\_Sch\_EV_{it} = idle$
13:   else
14:     set $New\_Sch\_EV_{it} = Sch\_EV_{it}$
15:   end if
16: end for
```

Furthermore, the algorithm checks if the State of charge for an EV has reached its maximum limit if it has and the forecasting model is signalling to still charge the vehicle a new schedule is set to idle for that vehicle. If none of the condition is met then the new schedule is set to the one forecasted by the machine learning model.

4 Evaluation & Analysis

We simulated a scenario with ten EVs used to reduce the peak demand of one commercial building. In the simulation we had recorded the building energy consumption at a timestep of 15 minutes, along with details of each EV such as vehicle ID, state of charge of vehicle, minimum state of charge for that vehicle, maximum state of charge, transfer rate, time of arrival and time to leave. Based on these data our LEMS creates a charging/discharging schedule for each connected EV – along with a log file that contains values for each of the input paramters. We developed a forecasting model using the Weka toolkit. To identify the most appropriate classification algorithm to use for developing the model, we considered both generative and discriminative techniques to identify the most appropriate classifier. For generative models we looked at models that consider conditional dependencies (Bayesnet)[34] and those that looked at conditional independence (Naivebayes)[37] in the dataset. For discriminative model we considered decision trees (J48)[35] and a neural network (multilayer perceptron)[36].

Machine Learning Model (MLM)	**Accuracy of MLM**	**Accuracy After Applying Rule Based Component**
NaiveBayes	76%	82%
BayesNet	85%	87%
J48	82%	83%
Multilayer Perceptron	80%	83%

Table 1: Accuracy of Schedule Forecasting Algorithm at Edge

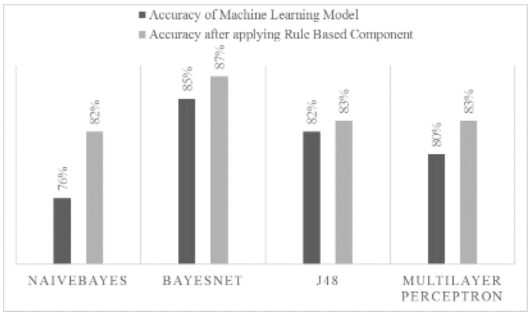


Fig. 3: Accuracy of Machine learning model vs Overall Accuracy of model deployed at Network Edge

While each model trained on the dataset to identify the best classifier we used the standard/default configuration for our classifier. Once the model was trained we created a log file to simulate the a scenario in which communication link was broken between the LEMS and the charging station and an EV had arrived after the communication failure. This log file was used as the testing dataset for the model. The log file considered arrival of the EV at different times with different states of charge. Based on their state of charge, time of day, energy consumption of building a charging schedule was forecasted. The accuracy of each model is shown in the table below. We can see from the table that the Bayesnet performed the best (85%) among the four models we created, which showed that data had conditional dependency and NaiveBays performed worst (75%). Once the schedule was created, the rule based component compares the scheduling decisions made for the connected EV and updates the schedule. As we can see the accuracy of the forecasting model increased once the rules were applied. From Table 1 and figure 3, we can observe that the combination of Bayesian model and rule based component gave the highest accuracy of 87%.

5 Conclusion

We investigated the integration of a Local Energy Management System (LEMS), which is cloud hosted and needs to acquire data over a network to develop a charging/discharging schedule for EVs/ESUs, with capability that is located at the network edge. The LEMS system also made use of EV batteries/ESUs as an energy store, to flatten the peak energy demand of a building or to participate in an ancillary market. We looked at two scenarios in which there is a communication failure between the LEMS and that of charging stations. One scenario when no new EVs arrive at the buidling site is handled by developing a day-ahead schedule using the cloud-based LEMS and stored at the charging point. This schedule is then updated after each time step (set to 15 minutes in our simulation), and able to take account of any changes that might have occured in the system over this time period. In the second scenario we looked at a situation when a new EV arrives at a building site after a communications failure has occured between the LEMS and the charging point/gateway. In this situation we introduce a schedule generation capability that comprises of two components: (i) the LEMS-generated schedule and (ii) a rule-based component that is able to adapt the schedule based on locally recorded data. We tested our forecasting model by simulating a scenario in which an EV arrives at a site during a communication failure and found that our forecasting model can forecast the schedule for the new EV with

87% accuracy. Our approach demonstrates how a forecasting model that has been generated at a data center can be combined with an adaptation mechanism that is located at the network edge, and able to adapt the forecasting model in case of network failure (between the data center and the edge) and taking account of latency considerations (i.e. having to transfer data about recently arrived EVs to the cloud-based LEMS to update the schedule). The general approach described here can also be generalised to other similar types of applications.

Acknowledgment

This work was carried out in the InnovateUK/EPSRC-funded "Ebbs and Flows of Energy Systems" (EFES) project.

References

1. I. E. Agency. Energy efficiency. `http://www.iea.org/about/faqs/energyefficiency/`, 2017. (Accessed on 01/11/2017).
2. K. Amarasinghe, D. Wijayasekara, and M. Manic. Neural network based downscaling of building energy management system data. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 2670–2675. IEEE, 2014.
3. S. Bera, S. Misra, and J. J. Rodrigues. Cloud computing applications for smart grid: A survey. *IEEE Transactions on Parallel and Distributed Systems*, 26(5):1477–1494, 2015.
4. F. Bonomi. Connected vehicles, the internet of things, and fog computing. In *The Eighth ACM International Workshop on Vehicular Inter-Networking (VANET), Las Vegas, USA*, pages 13–15, 2011.
5. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.
6. J. Diaz-Montes, M. AbdelBaky, M. Zou, and M. Parashar. Cometcloud: Enabling software-defined federations for end-to-end application workflows. *IEEE Internet Computing*, 19(1):69–73, 2015.
7. J. Diaz-Montes, Y. Xie, I. Rodero, J. Zola, B. Ganapathysubramanian, and M. Parashar. Exploring the use of elastic resource federations for enabling large-scale scientific workflows. In *Proc. of Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS)*, pages 1–10, 2013.
8. Z. M. Fadlullah, D. M. Quan, N. Kato, and I. Stojmenovic. Gtes: An optimized game-theoretic demand-side management scheme for smart grid. *IEEE Systems journal*, 8(2):588–597, 2014.
9. M. Giacobbe, A. Celesti, M. Fazio, M. Villari, and A. Puliafito. A sustainable energy-aware resource management strategy for iot cloud federation. In *Systems Engineering (ISSE), 2015 IEEE International Symposium on*, pages 170–175. IEEE, 2015.
10. R. C. Green, L. Wang, and M. Alam. Applications and trends of high performance computing for electric power systems: Focusing on smart grid. *IEEE Transactions on Smart Grid*, 4(2):922–931, 2013.
11. K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe. Mobile fog: A programming model for large-scale applications on the internet of things. In *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, pages 15–20. ACM, 2013.
12. L. Ji, W. Lifang, and Y. Li. Cloud service based intelligent power monitoring and early-warning system. In *Innovative Smart Grid Technologies-Asia (ISGT Asia), 2012 IEEE*, pages 1–4. IEEE, 2012.
13. H. Kim, Y.-J. Kim, K. Yang, and M. Thottan. Cloud-based demand response for smart grid: Architecture and distributed algorithms. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 398–403. IEEE, 2011.
14. D. Korzhyk, V. Conitzer, and R. Parr. Solving stackelberg games with uncertain observability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1013–1020. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
15. J. Laustsen. Energy efficiency requirements in building codes, energy efficiency policies for new buildings. *International Energy Agency (IEA)*, pages 477–488, 2008.
16. X. Li and J.-C. Lo. Pricing and peak aware scheduling algorithm for cloud computing. In *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*, pages 1–7. IEEE, 2012.
17. Z. Li and M. Parashar. A computational infrastructure for grid-based asynchronous parallel applications. In *Proceedings of the 16th international symposium on High performance distributed computing*, pages 229–230. ACM, 2007.
18. B. Lohrmann and O. Kao. Processing smart meter data streams in the cloud. In *Innovative Smart Grid Technologies (ISGT Europe), 2011 2nd IEEE PES International Conference and Exhibition on*, pages 1–8. IEEE, 2011.

19. H. Lv, F. Wang, A. Yan, and Y. Cheng. Design of cloud data warehouse and its application in smart grid. In *Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on*, pages 849–852. IET, 2012.
20. H. Madsen, B. Burtschy, G. Albeanu, and F. Popentiu-Vladicescu. Reliability in the utility computing era: Towards reliable fog computing. In *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 43–46. IEEE, 2013.
21. S. Maharjan, Q. Zhu, Y. Zhang, S. Gjessing, and T. Basar. Dependable demand response management in the smart grid: A stackelberg game approach. *IEEE Transactions on Smart Grid*, 4(1):120–132, 2013.
22. J. D. Montes, M. Zou, R. Singh, S. Tao, and M. Parashar. Data-driven workflows in multi-cloud marketplaces. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 168–175. IEEE, 2014.
23. T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam. Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. In *Proceedings of the first international workshop on Mobile cloud computing & networking*, pages 19–26. ACM, 2013.
24. N. I. of Standards and Technology. Cyber-physical systems — nist. `https://www.nist.gov/el/cyber-physical-systems`, 2016. (Accessed on 01/11/2017).
25. B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran. Migcep: operator migration for mobility driven distributed complex event processing. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 183–194. ACM, 2013.
26. L. Pérez-Lombard, J. Ortiz, and C. Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.
27. U. N. E. Programme. Why buildings. `http://www.unep.org/sbci/AboutSBCI/Background.asp`, 2016. (Accessed on 01/11/2017).
28. T. Rajeev and S. Ashok. A cloud computing approach for power management of microgrids. In *Innovative Smart Grid Technologies-India (ISGT India), 2011 IEEE PES*, pages 49–52. IEEE, 2011.
29. S. Rusitschka, K. Eger, and C. Gerdes. Smart grid data cloud: A model for utilizing cloud computing in the smart grid domain. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 483–488. IEEE, 2010.
30. A. Sheikhi, M. Rayati, S. Bahrami, A. M. Ranjbar, and S. Sattari. A cloud computing framework on demand side management game in smart energy hubs. *International Journal of Electrical Power & Energy Systems*, 64:1007–1016, 2015.
31. Y. Simmhan, S. Aman, A. Kumbhare, R. Liu, S. Stevens, Q. Zhou, and V. Prasanna. Cloud-based software platform for big data analytics in smart grids. *Computing in Science & Engineering*, 15(4):38–47, 2013.
32. I. Stojmenovic and S. Wen. The fog computing paradigm: Scenarios and security issues. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pages 1–8. IEEE, 2014.
33. L. Tang, J. Li, and R. Wu. Synergistic model of power system cloud computing based on mobile-agent. In *Network Infrastructure and Digital Content (IC-NIDC), 2012 3rd IEEE International Conference on*, pages 222–226. IEEE, 2012.
34. University of Waikato. Bayesnet. `http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/BayesNet.html`, 2017.
35. University of Waikato. J48. `http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html`, 2017.
36. University of Waikato. Mlpclassifier. `http://weka.sourceforge.net/doc.packages/multiLayerPerceptrons/weka/classifiers/functions/MLPClassifier.html`, 2017.
37. University of Waikato. Naivebayes. `http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/NaiveBayes.html`, 2017.
38. C. Wei, Z. M. Fadlullah, N. Kato, and I. Stojmenovic. On optimally reducing power loss in micro-grids with power storage devices. *IEEE Journal on Selected Areas in Communications*, 32(7):1361–1370, 2014.
39. T. Weng and Y. Agarwal. From buildings to smart buildingssensing and actuation to improve energy efficiency. *IEEE Design & Test*, 29(4):36–44, 2012.
40. D. Wijayasekara, O. Linda, M. Manic, and C. Rieger. Mining building energy management system data using fuzzy anomaly detection and linguistic descriptions. *IEEE Transactions on Industrial Informatics*, 10(3):1829–1840, 2014.
41. D. Wijayasekara and M. Manic. Data-fusion for increasing temporal resolution of building energy management system data. In *Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE*, pages 004550–004555. IEEE, 2015.
42. N. Yaagoubi and H. T. Mouftah. User-aware game theoretic approach for demand management. *IEEE Transactions on Smart Grid*, 6(2):716–725, 2015.
43. C.-T. Yang, W.-S. Chen, K.-L. Huang, J.-C. Liu, W.-H. Hsu, and C.-H. Hsu. Implementation of smart power management and service system on cloud computing. In *Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2012 9th International Conference on*, pages 924–929. IEEE, 2012.