A simulation model for the control of electric vehicles charging networks

Davydova, A., Lupin, S., Vagapov, Y. and Chen, Z.

A Simulation Model for the Control of Electric Vehicles Charging Networks

Anastasia Davydova, Sergey Lupin
National Research University of Electronic Technology,
Zelenograd, Moscow, Russia

Yuriy Vagapov, Zheng Chen
Glyndwr University, Plas Coch, Mold Road,
Wrexham, LL11 2AW, UK

***Abstract*—This paper presents the details of software development and implementation of a simulation model to be used for performance analysis of the control applied for management of a plug-in electric vehicles charging network. The model is designed to provide an evaluation of the tested control algorithms in terms of efficiency and stability operation. Apart from the control algorithm evaluation the model can be used for assessment of required control modification. The model has been built using Java language under paradigms of object-oriented programming and test driven development.**

Keywords—simulation model; electric vehicle; energy management system; charging station; energy demand

I. Introduction

A growing number of plug-in electric vehicles in use requires expansion of the battery charging infrastructure connected to power grid. Uncontrolled charging through the stations located in urban areas can bring an impact on electric power system and reduce quality of electricity and reliability of power distribution [1], [2]. Therefore the charging points and stations in a certain area have to be integrated into a network to be operated under a control algorithm in order to avoid power grid overloading. The network control also involves eclectic vehicles owners directing them to the charging points for the fastest and efficient battery recharging.

The control strategy of the charging network is based on multi-criterion optimisation where the control algorithm efficiency and stability are crucial. These parameters can be assessed using a model simulation approach. This paper describes a model of management system for plug-in electric vehicles charging network which is used as a tool for evaluation of stability of the control algorithm.

II. Control System Algorithmic Stability

Fig. 1 shows a typical control scheme where CO is a control object and CD is control device operating under a known algorithm. CD collects information about the environment from a finite number of sensors $x_i,(i = 1,N)$. Generally, the main task of control device is to provide a state of the control object where a certain target function is equal to the function $y(t)$. At the same time CD is able to control the current value of the target function $y^*(t)$ and its deviation from the ideal $\Delta = y - y^*$. There can exist few target and criterion functions in the real systems.

To assess the stability of such control system it is necessary to go through all the possible sensors states. The digitised output from each sensor can take 2^k values, where k is a number of bits. Then the number of all possible states of the system is $(2^k)^N$, where N is a number of sensors. Thus, the stability analysis problem becomes computationally complex at $k = 10$ and $N = 3$. However in real systems, the dimensions of these parameters are significantly higher. A special feature of the problem comprises in the fact that the majority of $(2^k)^N$ system states cannot be realised in practice, but it may become the input signals combination for CD in case of error.

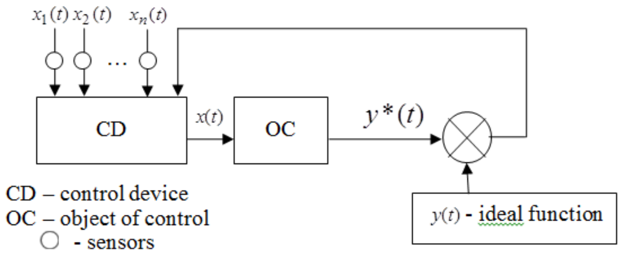

Fig. 1 Control system with a feedback and input sensors.

In the traditional approach to analysis of the control system performance, the systems are checked only within physically realisable range of the sensor signals (boundary test method). However, this approach assumes that the sensor readings correctly indicate the state of the environment and the system as a whole.

Another approach is based on the fact that the underlying CD algorithm utilises not all the information about the system status for generating the control action. In our case it means that there exists a group of $(N - M)$ sensors x_i, $(i = N + 1,M)$, which readings are not used by the control device to develop the response. However, the environment monitored by such sensors will indirectly effect on the readings of other sensors and, therefore, influence the stability of the control system. The same approach is applicable for the system having a faulty sensor.

The task of evaluating algorithmic stability can be summarised as following: it is necessary to analyse the correctness of the generated control actions developed by CD when distortion of the sensor readings occur. The level of distortion is a parameter of the analysis and another property of the control system. [3].

III. CONTROL SYSTEM SIMULATION MODEL

Research on the algorithmic stability were carried out using an example of a charging stations network for plug-in electric vehicles (EV). The criterion for the control stability in this example is the ability of application distribution algorithm to make informed decisions in case of distortion of the input information.

There are the following main parameters (and its values) in the research model:

- number of charging stations – 3;
- number of sockets at each charging station – 5;
- number of active EVs – 200;
- number of control system sensors – 15;
- application distribution algorithm – the nearest available station, the least loaded station, an intelligent algorithm;
- criterion of the decision correctness is a queue length;
- variable parameters – level of error signals from the sensors;
- number of model simulations with the same parameters S = 100.

There are 3ways to control electric cars:

- centralised control strategy;
- decentralised control strategy;
- absence of control.

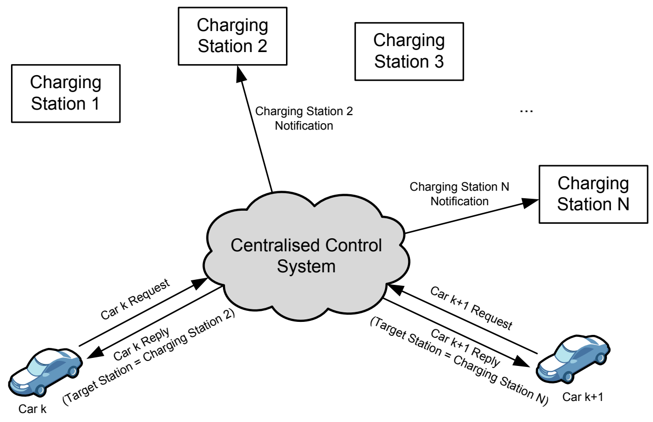
Charging Station 1
Charging Station 2
Charging Station 3
...
Charging Station 2 Notification
Charging Station N Notification
Charging Station N
Centralised Control System
Car k Request
Car k Reply (Target Station = Charging Station 2)
Car k+1 Request
Car k+1 Reply (Target Station = Charging Station N)
Car k
Car k+1

Fig. 2. Scheme of load distribution between the nodes of the charging station network using centralised control strategy.

Fig. 2 describes the principle of a centralised control system used for load balancing. In this case, when the owner of an EV recognises the need to recharge battery of EV, it sends a request to the centralised system (Car k Request). The system has all required information about stations load and selects the optimal charging station for EVs. The answer to request contains information about the target charging station (Car k Reply, Target Station = Charging Station 2). In order to simplify the implementation it is assumed that each EV directed to the charging station goes to the appointed charging station. Therefore, the process of confirmation is not considered here. After the response, the control system sends a notification to the appointed charging station.

Fig. 3 shows a decentralised method of management when the owner of each vehicle makes a decision to which charging station to go on the basis of responses from the charging stations. The vehicles send request to the charging stations (Car k Request, Car k+1 Request). Each station sends information about its load to the client (Charging Station N Reply). Based on this information the owner selects the charging station to go and send notification to it.

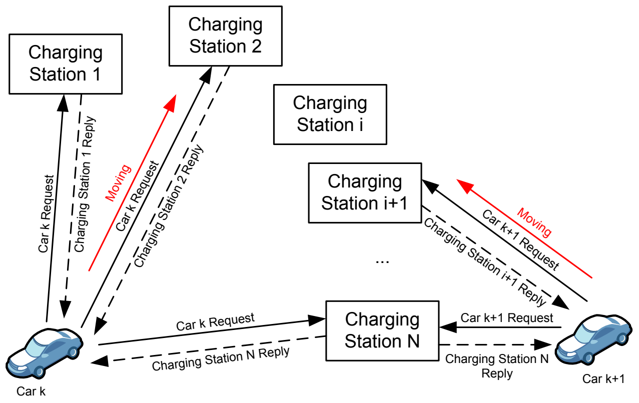

Fig. 3. Scheme of load distribution between the nodes of the charging station network using decentralised control strategy.

Centralised control strategy utilises Reliable Fetch Algorithm (RFA) for cars distribution. The algorithm operation is based on information obtained from video cameras and other additional sensors. The outcome of the algorithm processing is the advice to a EV owner which charging station has been selected for the battery refilling. This algorithm assigns initial minimal queue length to the length of 1st dispatcher queue and then calculates 2 values to evaluate queue length for each dispatcher

$$L_{K1} = dispenser.getVideoCamera()Length().$$

$$L_{K2} = L_{\Sigma} - \sum\nolimits_{i=1, i \neq i}^{n} dispensers[i].getVideoCamera().Length()$$

In case that $L_{K1} == L_{K2}$ and $L_{K1} < L_{min}$ then $L_{min} = L_{K1}$. If $L_{K1} \neq L_{K2}$ then average value is calculated by $L_K = (L_{K1} + L_{K2})/2$. If $L_K < L_{min}$ then $L_{min} = L_K$.

Decentralised control strategy uses an assumption that the majority of EV owners will prefer a station located closer to them. In the simulation model the third charging station is located a little further than the other two in order to investigate how it will affect the uniformity of EV distribution between the stations.

IV. SOFTWARE IMPLEMENTATION

The simulation model has been build Java programming language under paradigms of Object-Oriented Programming and Test Driven Development (TDD) [4].

TDD concept assumes developing software in short iterations, starting with the writing unit tests, and then writing code that ensures completion of all written tests. During that step the tests and the code are refactored [5]. Thus, a TDD ensures that the code base is easy to test, these tests are simple and reflect the most current documentation for the system.

The test module of the application enables flexible modification of the business logic because a high level of code coverage guarantees timely identification of emerging errors. The consistent state of the program assumes 100% successful completion of all implemented and included in the assembly tests.

The main domain classes of the model are *Model*, *Car*, *ChargingStation*, *Dispenser*, *VideoCamera* and *CarState* (Fig. 3). Class *Model* is the root class and includes references to instances of classes *Car* and *ChargingStation* describing entities of the car and charging station. The charging station includes a set of sockets (*Dispenser*) with installed surveillance camera (*VideoCamera*).

Object *Car* can exist in the following 6 states (Fig. 4):

- the target charging station is not defined (*NoTargetState*);
- movement to the target charging station (*MovingToStationState*);
- the car arrived at the station (*AtStationState*);
- waiting in queue at the station (*WaitingInQueueState*);
- in the process of charging (*ChargingState*);
- the car is charged (*ChargedState*).

The transition from one state to another is performed within the *Car* class methods, as well as in the process of charging, for which *Dispenser* is responsible.

In addition to the above major classes models there also were implemented a number of additional classes in which the main simulation loop works. It provides generation of cars flow during the day, initialisation of their original state, the generation of requests for charging, time simulation and objects movement. The main class that implements the logic simulation model is *DynamicSimulator* and it is inherited from *AbstractSimulator*. It manages *Car*, *Request* (application of an electric vehicle to charge) objects, *Printer* (writes the results of simulations in the text file) and class *StationsMonitor*, responsible for the distribution of requests between the charging station (via *IFetchAlgorithm* interface implementations) and the collection of statistics from them.

The simulator provides modelling of the daily work of charging stations control system when the randomly generated requests for charging enter the system. The initiation of the simulation process is carried out in *DynamicSimulatorTest* test class where the initial settings and the number of simulations are configured. Also this class counts the number of successfully serviced requests according to the different distribution strategies requests between nodes of the charging station network.

The simulation model allows testing the control algorithms of application distribution on the stability using simulation of incorrect information submission to the control system input. This may be, for example, information about the length of the queue at the charging stations. Since the queue length is one of the parameters of the decision algorithm, the wrong decision would be taken in case of input data corruption. Which in turn leads to an inefficient management.

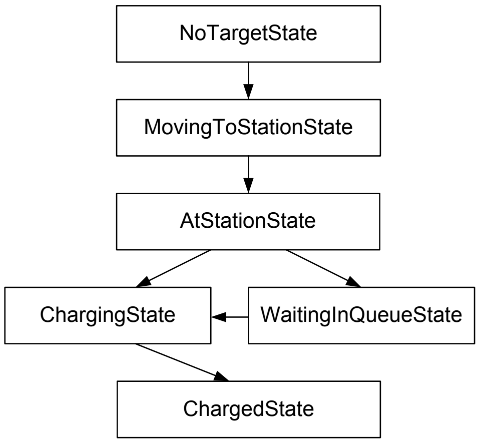

Fig. 4 The state diagram of management system for EV charging network.

It should also be noted that the model conforms to the principles of object-oriented programming: inheritance, encapsulation and polymorphism. The functional common for the related functional objects was localised in the base classes. All interaction between the components was implemented using interfaces, not at the class level. The implementations of the necessary overloaded methods were done in the derived classes.

V. Simulation Results

The behaviour of the EVs was analysed using a set of S simulations with fixed parameters. Then, the numbers of served EVs have been summarised for each set of simulating parameters. Using the obtained data, the mathematical expectation is estimated and the average number of serviced EVs per 24 hours for chosen strategy is calculated as following:

$$N_{charged} = \frac{1}{S}\sum_{i=1}^{N} N_{chargedi} \tag{1}$$

In the simulations for this research S was taken equal to 100.

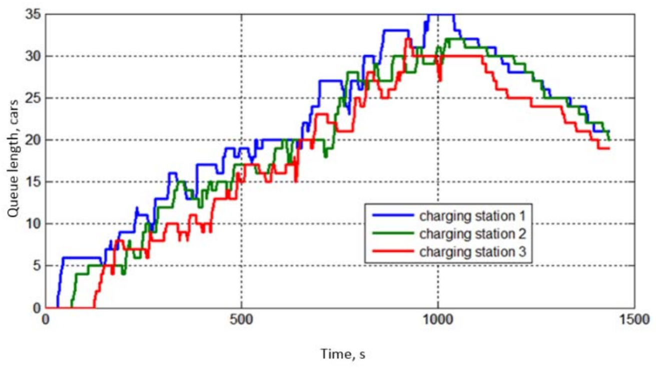

Fig. 5 Total queue lengths at the charging stations 1, 2, 3 for the centralised distribution strategy.

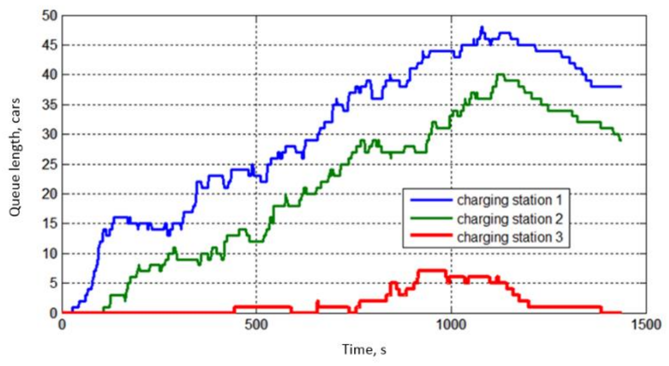

Fig. 6 Total queue lengths at the charging stations 1, 2, 3 for the decentralised distribution strategy with the preference for nearest station.

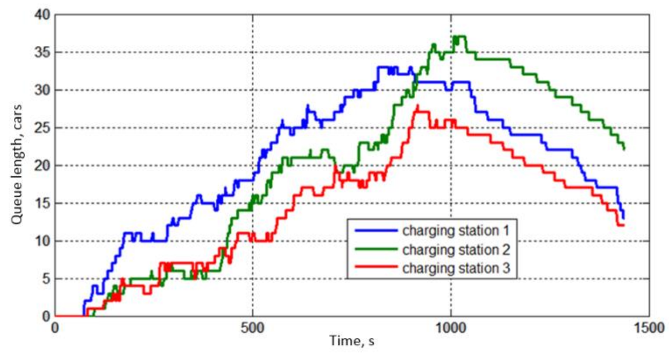

Fig. 7 Total queue lengths at the charging stations 1, 2, 3 for the decentralised distribution strategy without preference to any station.

This work focus on comparison of centralised and decentralised distribution strategies. Simulation results shown in Fig. 5-7 demonstrate that the most uniform distribution of EVs is characteristic for centralised control strategy.

It has been observed that the results of simulation for decentralised distribution strategy show also quite evenly distribution. However, using decentralised strategy with the preference for nearest stations leads to the growth of queues length at these stations while the third one is almost not used.

VI. CONCLUSIONS

In order to investigate performance of the control of an eclectic vehicle charging network a simulation model comprising multi-criterion optimisation algorithm has been developed. The model algorithm is focused on generating the optimum solution for all objects of the model including charging points and stations and electric vehicle owners. The solution is actually dynamic depending on a large number of external parameters such as a number of electric vehicles travelling in the area, their battery state of charge status, power grid energy demand, etc.

The investigation of different distribution strategies (centralised and decentralised) shows advantages of centralised control algorithm which distributes load evenly between stations and prevents queue length growth at some charging stations. But this conclusion is only applied for the intellectual algorithms of centralised distribution which consider current load at each station and uses different sources of information (different types of sensors).

Java programming language has been used for the model software design and implementation under paradigms of object-oriented programming and TDD. The implemented model runs an algorithm under investigation through a series of simulations where external parameters are fixed for each test. Then, the statistics obtained from the simulations is collected for the model objects (e.g. charging stations, serviced vehicles) and evaluated in terms of efficiency and stability operation. The simulation model has been proved to be an effective tool for analysis of existing management strategies. The model can be also used for assessment of required modifications of the control demanding, for example, installation of additional sensors to the system.

REFERENCES

[1] R. Faria, P. Moura, J. Delgado, and A.T. De Almeida, "Managing the charging of electrical vehicles: Impacts on the electrical grid and on the environment," *IEEE Mag. Intelligent Transportation Systems*, vol. 6, no. 3, pp. 54-65, 2014.

[2] M. Bucher, Y. Vagapov, A. Davydova, and S. Lupin, "Estimation of electrical energy demand by electric vehicles from households: A UK perspective," in *Proc. IEEE Conf. Young Researchers in Electrical and Electronic Engineering*, St. Petersburg, Russia, Feb. 2-4, 2015, pp. 159-164.

[3] A. Davydova, S. Lupin, and Y. Vagapov, "Assessment of survivability of complex control systems using simulation methods," in *Proc. 12th IEEE East-West Design and Test Symposium EWDTS 2014*, Kiev, Sept. 26-29, 2014, pp. 213-216.

[4] Java Standard Edition (14 Jan. 2014) [Online] Available: http://www.oracle.com/technetwork/java/javase/overview/index.html

[5] I. Graham. *Object-oriented Methods: Principles and Practice*, 3rd ed., Moscow: Williams, 2004.