SENITY: An Emulator for Smart Energy Management of Households in a City

Kyriaki Seklou
Department of Informatics and Telecommunications
University of Peloponnese
Tripoli, Greece
kseklou@uop.gr

Panagiotis Kokkinos
Inst. of Comm. & Computer Sys.
ICCS - NTUA
& Dep. of Digital Systems
University of Peloponnese
Greece
p.kokkinos@uop.gr

Vassilis Poulopoulos
Department of Digital Systems
University of Peloponnese
Sparta, Greece
vacilos@uop.gr

Nikolaos D. Tselikas
Department of Informatics and Telecommunications
University of Peloponnese
Tripoli, Greece
ntsel@uop.gr

***Abstract*—Energy is essential for most activities in our modern society. As a result, energy production, storage, transfer and consumption are key issues, considered both in academic and in industry. Today, a number of global events make more than necessary the need for the wide adoption of simple practices in order to reduce energy consumption in households. These practices if applied can significantly reduce the energy footprint of large cities and of MegaCities. In our work, we introduce the SENITY emulator through which it is possible to evaluate various energy consumption practices both for research and for educational purposes. We present the architecture of SENITY, its implementation details, and simple interactions through SENITY's web user interface.**

***Index Terms*—energy consumption, households, city, emulator**

I. Introduction

The adoption of renewable energy sources against fossil fuels so as to reduce greenhouse gas emissions, the exploitation of the Information and Communication Technologies (ICT) towards making the energy grids more secure, reliable, efficient and flexible [1] and the liberalization of the electricity markets with new business actors/models are some of the factors that revolutionize the energy sector.

Today, energy production, storage and consumption are key issues in national and international level both for environmental reasons but also for economic ones. What is more, the COVID-19 pandemic and the war in Ukraine have disrupted the global energy market, leading to rising energy prices with domino effects in cost-of-living and social stability. Authorities are urging for the adoption of energy practices in households, such as the reduction of the air-conditioners' (A/Cs) temperature of operation.

In 2020, households represented 27% of final energy consumption in the EU [2]. Households use energy for various purposes: space and water heating, space cooling, cooking, lighting and electrical appliances and other end-uses. Interestingly, the appearance of more and more smart devices provides opportunities for the adoption of remote and intelligent energy consumption reduction practices either in individual level or in a city level. This requires that all citizens follow suggested well-practices and that centralized mechanisms are being used for the enforcement of such practices, e.g., by turning on or off home appliances based on the energy production or the current cost of energy [3].

This problem becomes even more interesting in MegaCities. A Megacity is, according to the definition of the United Nations, a city with more than 10 million inhabitants. By 2030, the world is projected to have 43 Megacities, most of them in developing regions [4]. Megacities are notable for their size and concentration of economic activity. The economies and societies of Megacities have developed rapidly, but they are also under pressure from increasing population, resources, and ecological threats. Sustainable energy supply is one of the main challenges in Megacities [5].

SENITY (Smart ENergy In The citY) is an emulator for smart energy deployments targeting city environments that can assist in the investigation of some of the mentioned issues. SENITY can emulate cities that consist of thousands of households that contain appliances able to report their energy consumption. These appliances are also manageable, meaning that one can turn them on or off and/or adjust other parameters of their operation (e.g., temperature) [6]. SENITY has been designed as a multi-role platform. So, on the one hand it can be used as a research tool for researchers developing energy consumption models, load balancing practices, pricing policies and other mechanisms interacting with SENITY through the provided Application Programming Interface (API). On the other hand, it can also be used as an educational tool, which can showcase through gamification, simple ways in which a city's energy consumption can be reduced, following simple - every day practices. The latter can be achieved by utilizing SENITY's web user interface (UI).

The remainder of this paper is organized as follows. In Section II, we present SENITY's architecture. Section III describes the format and the capabilities of the models used by the emulator. Section IV provides implementation details. In Section V, we illustrate basic ways to use SENITY through its UI. Future work and conclusions are presented in Section VI.

979-8-3503-9958-5/22/$31.00 ©2022 IEEE

II. Architecture

SENITY's emulator architecture is presented in figure 1.

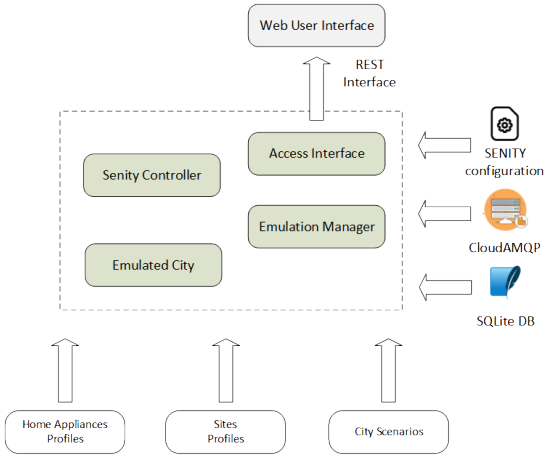

Fig. 1. SENITY's architecture details.

SENITY has the following main components:

- Senity Controller: It is the core component of the SENITY emulator, which initiates the Access Interface and the Emulation Manager.
- Access Interface: It realizes the SENITY REST-based Application Programming Interface (API) and passes information and commands to the SENITY Controller.
- Emulation Manager: It handles all the aspects of the emulation regarding the creation of a new city and its operation.
- Emulated City: It is the component that emulates a particular city, calculates the energy consumption and reacts to commands regarding the city's operation.
- Web User Interface: It is a web-based interface that interacts with SENITY through the provided API and the database.
- SENITY configuration: This is a JSON file that holds configuration information regarding the operation of the SENITY emulator.
- CloudAMQP: The CloudAMQP service is used to provide cloud-based messaging broker service for the communication of the SENITY component.
- SQLite DB: This is a simple db that is being used for storing information like a city's status (e.g., which devices are on and off), the current consumption and other.
- Profiles: A set of JSON files define the characteristics of the appliances, the sites and overall of the emulated city (Section III).

III. Energy and City Models

An important aspect of the SENITY emulator is the files that describe the devices, the households (sites) and the cities. As reported in Section II, these files follow a JSON-structured schema. The characteristics of the emulated city, the energy footprint of the city and the capabilities for dynamic management of the household devices are based on the definition provided in these files. In what follows we present some simple examples of these files.

In a household device JSON file such as in listing 1, the *avgConsumption* is the device's consumption in watts per minute, while the *workingCycle* indicates whether the device operates all the time in a 24/7 manner or in shorter working cycles every time it is invoked (e.g., 5/24). *initialStatus* indicates whether the device is by default on (1) or off (0) when the emulation is started. *updateInterval* is the interval in which measurements are sent by the device, and override the per site *updateInterval*. As future work, more complex consumption profiles (e.g., based on the temperature of operation of the washing machine) will be investigated. In the current version of SENITY a number of device profiles are provided for A/Cs, washers, fridges and ovens. It is possible to define different device profiles for the same type of device, for example, an A/C with inverter and an A/C without one. This is an interesting example since one can evaluate through SENITY the effect of using inverter-based A/Cs in the energy consumption of a city.

```
{ "name" : "aircondition01",
  "type" : "aircondition",
  "initialStatus": 1,
  "updateInterval": 10,
  "avgConsumption": 1100,
  "workingCycle": 5
}
```

Listing 1. A household's device json file

A site profile file defines a particular site and the number of its household devices. As for example, site01 in listing 2 has an oven, a fridge, three A/Cs and a washer.

```
{ "siteName": "site01",
  "devicesAvailable":
  [ {"deviceName": "oven01", "deviceCounter":
      1},
    {"deviceName": "fridge01", "deviceCounter"
       : 1},
    {"deviceName": "aircondition01", "
       deviceCounter": 3},
    {"deviceName": "washer01", "deviceCounter"
       : 1} ]
}
```

Listing 2. A site profile file

An example of a scenario file that dictates the city parameters of the emulation executed is presented next. This illustrates a small city with 100 households (sites) of type site01 and 200 sites of type site02.

```
{  "scenarionName": "scen01",
     updateInterval   : 10,
  "sitesAvailable":
  [ {"siteName": "site01", "siteCounter": 100
     },
    {"siteName": "site02", "siteCounter": 200
       } ] }
```

Listing 3. A scenario file

IV. Implementation and Operation

SENITY is implemented in python and utilizes a number of typical packages (pika, CloudAMQP, pyqt, sqlite3 and flask).

A. Operational flow

The overall operation flow of SENITY is described below:

- Site and device configurations in JSON format are created.
- A scenario is defined in JSON, utilizing existing site and device configurations.
- The emulator is started specifying the SENITY configuration, the devices and sites profiles' JSON files, along with the city scenario file. This starts the SenityController component that initiates the AccessInterface and the EmulationManager.
- A command that controls the way emulation is performed is received through the REST API (Subsection IV-B) implemented by the Access Interface component. The commands for the creation of a new city and for changing a city's status (e.g., setting devices on/off) are sent to the Emulation Manager through the messaging broker. For each new city a Universal Unique Identifier (UUID) is assigned and used for identifying the particular city. Other commands for retrieving the status of a city and its current energy consumption are handled by the Access Interface by retrieving the respective information from the database.
- The Emulation Manager receives from the messaging broker commands for the creation of a new city and for changing its status. In the former case, the Emulation Manager starts the Emulated City in a new thread. In the latter case, the Emulation Manager informs the thread running the respective Emulated City of its new status.
- An Emulated City runs continuously, calculating periodically the energy consumption of the city taking into account the devices that are on or off, according to the initial status of the city or the status that has been set through the respective commands.

B. REST API

Table I summarizes all the supported methods for the REST-based interface of SENITY.

Regarding the 1st API command, an example JSON file for the creation of a new city is presented in listing 4. The "population" label provides an alternative way for specifying the number of sites/households in a city, while the "interval" label sets the clock period of an emulated city. For example, an "interval" equal to 5, means that 1 emulated minute is equal to 5 seconds. SENITY's response is shown in listing 5.

```
{
    "population": 10,
    "name": "kalamata",
    "interval": 5
}
```

Listing 4. A json file for the creation of a new city

```
{
    "interval": 5,
    "name": "kalamata",
    "new_city": 0,
    "population": 10,
    "uuid": "232bbca8-1ed1-4ae0-9a21-ca6accd8f
        3b3"
}
```

Listing 5. SENITY's response

#	HTTP verd	URI	Description
1	POST	/api/v1/senity/city	Create a new emulated city. POST city characteristics through a JSON file. The response includes the city's uuid.
2	GET	/api/v1/senity/city/consumption/uuid	Request the consumption of a particular city providing its uuid. The response provides the latest consumption measurements for this city.
3	GET	/api/v1/senity/city/status/uuid	Request information for a particular city providing its uuid. The response provides a JSON with the city's sites characteristics and the devices' status (e.g., on or off)
4	POST	/api/v1/senity/city/status/uuid	POST the status of the devices (e.g., on or off) in a particular city through a JSON file.

TABLE I
THE SENITY REST API

Listing 6 is an example of the JSON response for the 3rd API command that provides the list of sites in a particular city. The "on_off" label for each site shows the status of the devices. The "consumption" label provides information regarding the reported consumption of the devices that is provided through the respective JSON-based device profiles.

```
{
[{""id"": 0, ""type"": ""site02"", ""devices""
    : [""oven01"", ""fridge01"", ""fridge01"",
     ""aircondition01"", ""aircondition01"",
    ""aircondition01"", ""dishwasher01""], ""
    on_off"": [1, 1, 1, 1, 1, 1, 1], ""
    consumption"": [3000, 350, 350, 1100, 1100
    , 1100, 1200]}, {""id"": 1, ""type"": ""
    site02"", ""devices"": [""oven01"", ""
    fridge01"", ""fridge01"", ""aircondition01
    "", ""aircondition01"", ""aircondition01""
    , ""dishwasher01""], ""on_off"": [1, 1, 1,
     1, 1, 1, 1], ""consumption"": [3000, 350,
     350, 1100, 1100, 1100, 1200]}]
}
```

Listing 6. A JSON response with the list of sites in a particular city

The 4th command is about controlling the status of a city. This version of SENITY supports only one operation type "on_off_site" that makes possible to manage which devices are on and which are off, affecting in this direct way the energy consumption of a city. In listing 7, the status of the devices of particular sites with id "0" and "1" are set accordingly.

```
{"operation_type": "on_off_site",
 "id" : [0, 1],
 "on_off": [[0,0,1,1,1,1, 0], [1,1,1,1,0,1,0]]
  }
```

Listing 7. Status of the devices in a site

V. Web User Interface (UI) and Results

SENITY also features a web user interface (UI). The UI is designed in a minimal manner in order to illustrate the basic capabilities of SENITY and it utilizes the SENITY's API.

The creation of the city and its sites can be completed using the available web forms. Then, the list of the cities created, is available in the UI's initial page, where it is possible to directly check the data concerning the cities and edit the information for each city. Moreover, for each city it is possible to view detailed dynamic energy information in two different sections: The first section is a real time presentation of the consumption, while the second - and most important for the system testing - is the one that enables massive or individual update of the status of each household's device.

Figure 2 presents the information available for a city. The user can get information about the real-time consumption, in addition with information about the city's number of households, number of devices and total power consumption of each device. It is also possible to observe which devices are turned on or off in each household. The active devices are marked with green color while the inactive ones are marked with red.

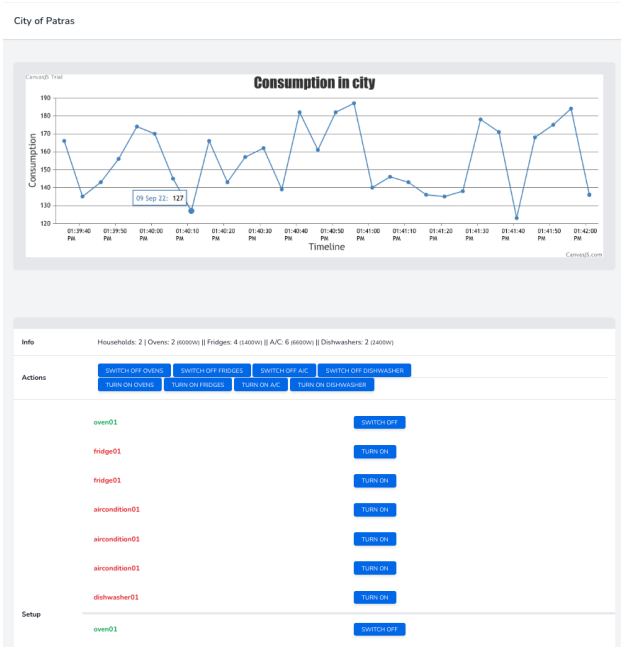

Fig. 2. SENITY UI city information.

The user can interact with the city, the households and the devices through buttons by massively turning on or switch off a certain type of device or particular devices in a specific household. The interaction is followed by the real-time "reaction" of SENITY, leading for example to results on the city's energy consumption. Figure 3 presents the typical energy consumption of a city when all households have their fridges turned on, while some of the other highly energy consuming devices are randomly selected to be active or inactive.

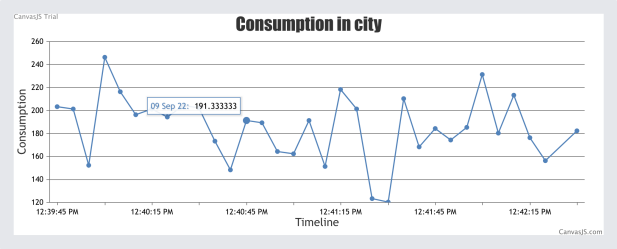

Fig. 3. A city's typical energy consumption (kwh).

Switching off a couple of A/Cs in a household may not have serious impact on the total consumption in a city, while deactivating all the A/Cs (e.g., on specific periods during a day) can considerably reduce the total energy consumption. Figure 4 presents what happens when a set of energy intensive devices are turned on in every household in a city.

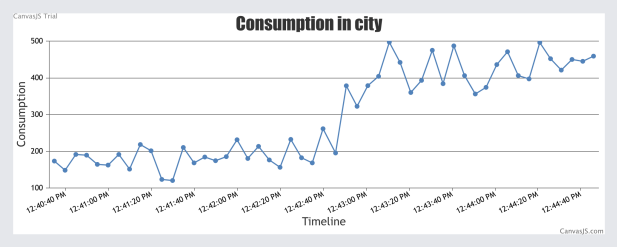

Fig. 4. Energy consumption when energy intensive devices are turned on.

VI. Conclusions

In this work, we presented SENITY, an emulator for the energy consumption of cities, concentrating on households and their appliances. We provided SENITY's architecture and implementation details and showcased its capabilities through the provided Application Programming Interface (API) and the web user interface (UI). Several extensions of SENITY are planned for advancing its capabilities as a research and a educational tool.

References

[1] https://www.un.org/en/chronicle/article/sustainable-development-goal-energy-and-information-and-communications-technologies

[2] https://ec.europa.eu/eurostat/en/web/products-eurostat-news/-/ddn-20220617-1

[3] E. O'Dwyer, I. Pan, S. Acha, N. Shah, Smart energy systems for sustainable smart cities: Current developments, trends and future directions, Applied Energy, Vol. 237, pp. 581-597, 2019.

[4] https://www.un.org/sustainabledevelopment/blog/2018/05/68-of-the-world-population-projected-to-live-in-urban-areas-by-2050-says-un/

[5] J. A. Paravantis, P. D. Tasios, V. Dourmas, G. Andreakos, K. Velaoras, N. Kontoulis, P. Mihalakakou, A Regression Analysis of the Carbon Footprint of Megacities, Sustainability, Vol. 13, No. 3, 2021.

[6] K. Seklou, P. Kokkinos, N. D. Tselikas, A. C. Boucouvalas: Monitoring and management of home appliances with NETCONF and YANG, Pan-Hellenic Conference on Informatics, (PCI), pp. 25–32, 2019.