# Snap4City: A Scalable IOT/IOE Platform for Developing Smart City Applications

C. Badii, P. Bellini, D. Cenni, M. Marazzini, P. Nesi, G. Pantaleo, M. Paolucci, M. Soderi, I. Zaza
*DISIT Lab, Department of Information Engineering – Università degli Studi di Firenze, Italy*
{name.surname}@unifi.it

E. G. Belay, F. Hachem, M. Mesiti, S. Valtolina
*Department of Computer Science – University of Milano, Italy*
{name.surname}@unimi.it

***Abstract*—Smart City solutions, initially started with open data, are evolving towards data aggregation and semantics. Recently, some of them are also offering IOT support. The combination of IOT and smart city is not an easy task, the data volumes are much higher than those addressed for industrial IOT. The complexity of IOT smart city solutions have been identified by a number of actors. The European commission started to set up the EIP project for stimulating and concerting actions. The Select4Cities project of the European Commission and associated community `http://www.select4cities.eu/` created a challenge to find research solutions satisfying a formalized set of functional and nonfunctional requirements. Snap4City presented in this paper is one of the solutions developed in response to that challenge. The solution proposed offers a platform where sophisticated IOT applications for controlling city dashboards as well as IOT mobile applications can be developed in few steps. Moreover, a number of development and monitoring tools have been developed. Among them, in this paper, a special attention is given to the tools and solutions for monitoring communication performance and to perform the assessment of scalability.**

***Index Terms*—IoT data, smart city, performance assessment.**

## I. Introduction

The world of ICT solutions for smart city is very wide encompassing services at different levels of complexity and coverage, from classical Open Data portals (CKAN [9], OpenDataSoft [19], ArcGIS and OpenData [2]) to sophisticate solutions that provide data aggregation and Smart City APIs facilitating the development of web and mobile applications (Km4City [5], City SDK [8]). The former solutions are mainly suitable for collecting and sharing open data files and their indexing on the basis of corresponding descriptive metadata. Open data, in those cases, can be uploaded by providing files in different formats: CSV, XLS, XML, SHP, etc. On the other hand, the latter solutions based on data aggregation and Smart City APIs try to create uniform data models describing the city entities and are focused on few domains (e.g. mobility and transport, culture and information, collection of feedback from the city users, e-heath). In more detail, most of the Smart City solutions provide to some extent: open data management, control room dashboards, data analytics, big data stores, data mining and data warehouse at the data ingestion, smart city API. Only few of them offer development tools for data analytics and application development [3].

The effectiveness of smart city services is enabled by the availability of solutions capable to combine public (open) and private data owned and managed by City Operators addressing specific domains (e.g. transport, mobility, energy, health, tourist operators, culture and events). These city stakeholders provide data and services of different volumes and at different time and space granularities. For example, in the city, we can have few energy operators with capillary house distribution and data collection, many public transport operators with thousands of vehicles/buses (millions of travels per year), some telecom operators with a very large number of mobile devices and millions of calls per year, and IOT networks of tens to hundred thousand of sensors with several millions of messages per year. Different granularity implies different methods for collecting and for providing access to data such as publication of open data files and/or statistics, real time publication of data updates in several different protocols for different purposes.

The IOT solutions arrived in the context of Smart City with new solutions deploying sensors/actuators with a set of protocols directly enforced in the devices, such as: OneM2M [20], ETSI M2M [15], MQTT [16], COAP, SigFox, AMQP, LoraWAN, Green Button Connect [11]. The IOT revolution has also opened the path for the integration of personal IOT devices and their exploitation in the context of personal applications; for example, for adding specific house temperature sensors and actuators, for managing personal health care sensors (e.g., glycosometer), training devices. Furthermore, the exploitation of contextual data coming from the city for taking personal decision is also very relevant. In most cases this issue is neglected and fully in charge of the single users. As a result, IOT solutions for smart city, providing support for the final users in exploiting city and personal data in the context of personal IOT applications, are really a few. See for example classical IBM, Google or AWS solutions that do not provide specific contextual and local data but facilitate the usage of your personal data only.

Considering the general scenario of IOT for Smart City, Select4Cities EC project and associated community (`http://www.select4cities.eu/`) launched a chal-

Select4Cities Consortium and the European Commission with Grant Agreement No:688196

lenge as a competitive PCP (Pre-Commercial Procurement) based on research activity according to the definition of EC. The challenge aimed to look for platforms that could cover a wide range of requirements at the same time in the context of IOT/IOE (internet of thing/everything) and smart city applications, with the aim of experimenting them on different cities in Europe such as Helsinki, Copenhagen and Antwerp. Among the identified requirements, there are the capability of the IOT/IOE solutions: to serve as a City Dashboard (this might mean to have centralized unique control room or to have several control rooms for each domain); to serve as an open city platform at service of different operators; to cope with real time communications of different kinds and formats; to manage data referential; and, to be capable to support the creation of a Living Lab enabling the city users and city stakeholders to develop their own applications by using different and suitable tools, and to put them at the disposal of the city. A set of challenges has been proposed by EIP European commission actions for Urban Platforms [10], mainly on clean power for vehicles, multimodality routing, smart logistic, sustainable mobility, etc.

In response to the above-mentioned research and development challenge, we have constituted the Snap4City team (`http://www.snap4city.org`) combining two research teams from the University of Firenze and University of Milano. The team has worked since the beginning of the 2017 for setting up the Snap4City platform. It is an open source IOT/IOE platform satisfying the above-mentioned requirements and a number of non-functional requirements regarding: open sources, scalability, standard compliance, robustness, distributed, managing heterogeneous communications, interoperability, security and privacy respect. At the present development stage, the first version of the Snap4City platform has been deployed and validated against several challenging cases as described in the rest of the paper, and it has been tested twice by the Select4Cities team. The solution proposed started from the exploitation of Km4City platform and its smart city API (`http://www.km4city.org`) [3], adding IOT/IOE capabilities and scalable management.

The paper is structured as follows. Next section discusses related work. Then, Section III presents the main requirements for IOT smart city solution and the proposed architecture. Section IV provides an example of what can be realized by using the Snap4City solution taking into account a mobility and transport scenario for infomobility. In Section V two scenarios of performance analysis for the proposed solution are presented. One is for the assessment of communications and the other one is for general scalability and workload. Conclusions are drawn in Section VI.

## II. Related Work

According to [12] more than 450 platforms for IOT have been presented that provide implementation for the functional blocks described in [4]. A comparison of the most representative IOT solutions at the state of the art is reported in Fig. 1. Among them, according to the information collected, we can see solutions that clearly declare to be suitable for the smart city scenarios and IOT such as Kaa, Bosch, FIWARE, CISCO, IBM and Carriots and few more. Among the graphical environments for the specification of applications working with streams of IOT data, NodeRed [18] is probably the most effective for rapid prototyping and has a large portability and limited footprint. Other solutions, as Eclipse, are based on Java and present larger footprint and more complex languages to be used. Moreover, NodeRed can be used in conjunction with Kafka, Spark, R studio, external services, etc., adding simple blocks, and thus the scalability on massive access to sensors can be delegated to data driven data collectors in blocks also exploiting IOT Directory/Discovery services.

| | Open Source end-to-end | Scalability IOT | Execution scalability | Visual Programming end-to-end applications | Advanced Smart City API | Multi Domain Semantic Platform | Process Alloc ation IOT API, scalability computing | Standard based Modules and IOT | City Dashboard H24/7 | Multi-protocol on IOT |
|---|---|---|---|---|---|---|---|---|---|---|
| *KAA* | Y | Y | Y | N | Y | N | Y | ? | N | Y |
| *IOT IGNITE* | Y (client) | Y | N | Y | N | N | N | N | N | MQTT only |
| *PTC ThingWorkx* | N | Y | Y | Y | N | N | N | Y | (via SQUEAL) | Y (via extensions) |
| *BEZIRK* | Y | N | N | N | N | Y | N | Y | N | Y |
| *Bosch IoT Suite* | N | Y | Y | Y | Y | N | N | Y | Y | Y |
| *FIWARE* | Y | (Y) | N | N | Y | N | N | Y | Y | Y |
| *CISCO Jasper* | N | Y | ? | N | N | N | N | N | N | N |
| *IBM Watson IoT* | N | Y | Y | Y (Node-RED) | Y | Y | N | Y | (Y) | Y |
| *Siemens MindSphere* | N | Y | ? | Y (via ATOS) | N | N | N | Y | Y (via IBM) | Y |
| *Carriots* | N | Y | ? | N | N | N | N | ? | Y | (MQTT only) |
| *Thingsboard* | Y | Y | Y | N | N | N | N | N | N | (MQTT, CoAP, http) |
| *IOT eclipse.ora* | Y | Y | ? | N | N | N | N | Y | N (via BIRT) | Y |

Fig. 1. Comparison among representative IOT platforms. Legend: *Y/N* present/not present feature; *(Y)* partially present; *?* no evidence from docs

Many efforts are nowadays devoted to the creation of virtual bridges among these platforms in order to guarantee the development of cross-platforms (also named horizontal) applications, that is applications able to connect sensors belonging to different platforms. European projects (like OpenIoT, BIG-IoT, Biotope, INTER-IoT, SymbIoTe) are moving in this direction and their idea is to offer facilities across all layers of the network stack in order to improve the interoperability among the different components involved in the management of sensors, actuators and network infrastructures. XGSN [6]

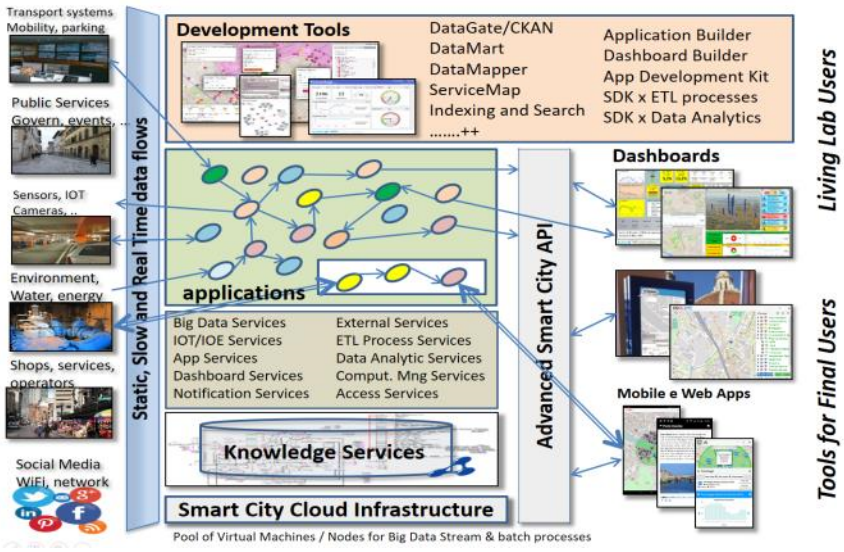



Fig. 2. Snap4City Architecture Overview

(extension of the GSN middleware [1]) is one of the first middleware (at the base of the OpenIoT project) for the IoT that supports a Domain Ontology for mitigating the semantic interoperability issues arising when integrating heterogeneous physical and virtual sensors. It exploits the SSN Ontology [7] for semantically annotating sensor data and observations in order to provide a standardized queryable representation that makes it easier to share, discover, integrate and interpret the data. Snap4city is moving in the same direction of this system with the following peculiarities. First, we adopt the Km4City Ontology that allows to better represent all the kinds of data that can be generated in a city (and not only the sensor/actuator data). Moreover, Snap4city adopts intelligent approaches for the semantic discovery of new sensors/actuators and their classification in the domain Ontology. Finally, Snap4city combines modern solutions in order to develop a scalable and efficient architecture that easily adapts to the millions of events that need to be treated in the context of smart cities.

A large number of smart city projects are focused on creating big data infrastructure and solutions such as REPLICATE H2020, RESOLUTE H2020, Triangulum H2020, EIP [10]. In [13] the case of smart city IOT integration has been discussed for the city of Santander without proposing details and performance analysis of the solution. In [14], a smart city IOT architecture has been proposed without addressing the aspects of scalability. As outlined in Fig. 1, none of the analyzed solutions is capable to address all the requirements. Most of them lack access to smart city data via API, many others are not scalable, a large number of them are limited in supporting multiple IOT protocols and data formats. We remark that the concept of scalability in the context of smart city is one of the most challenging with respect to those in the context of IOT for Industry 4.0 and agriculture applications due to the huge number of data flows related to the inhabitants.

## III. SNAP4CITY ARCHITECTURE

The main architecture of Snap4City consists of a set components developed for:

- Collecting data from open data, real time data, and IOT that are produced by different sources, city operators, and also by the users devices, social media.
- Storing and managing data in a knowledge base (KB) and tabular noSQL storages, and indexing them for data retrieval with inference, spatial-temporal reasoning, facet search, and drill down on time and space.
- Creating IOT applications, data driven and/or periodic, based on MicroServices [17]. The IOT Applications can be data flows extending NodeRED of IBM, which may also exploit Spark and Kafka capability, and personalized data analytics.
- Creating services/processes by means of a number of easily accessible tools for developing data analytics in R, Java, Python, etc., as well as for creating data transformation in ETL, Karma or NIFI.
- Executing and controlling in a reliable and scalable manner smart city processes (IOT, ETL and data analytic, exploiting data) on cloud infrastructure, that can be used to create smart city MicroServices for computing values periodically and/or in real time.
- Showing and navigating on data results via (i) city dashboards (for control room or for operators) for data drill down on time and space, and in turn facilitate the production of specific city dashboards for decision makers and city operators, at different levels; (ii) smart city API; and (iii) bulk data results.
- Providing access to data and services via (i) MicroServices for IOT and ETL applications, and via (ii) Advanced Smart City API for Web and Mobile App and dashboards.

Fig. 2 presents an overview of the Snap4City architecture, in which the data ingestion section has been implemented by using: (i) Km4City solution for ETL processes supporting protocols such as: OneM2M, ETSI, DATEX, Rest Call, FTP, Web Services, etc., mainly for ingesting data as Open Data, referral data, and real-time data provided by external services [3], (ii) a number of IOT Brokers to cover the connection with IOT devices, namely FiWare Orion Broker (NGSI protocol), Mosquitto (MQTT protocol) and RabbitMQ (AMQP protocol).

With the aim of abstracting the complexity of managing multiple IOT brokers and protocols, a new tool called IOT Directory has been developed in Snap4City. The IOT Directory allows registering IOT Brokers and collecting their corresponding registered devices, and propagating this information into the Km4City KB. The IOT Directory also provides support for IOT applications and processes that would like to subscribe to receive data driven from IOT devices, registered on the brokers. So that, when an IOT application is designed, the IOT application refers to the IOT directory/discovery to look for a device instead of searching for them manually on all single brokers. For example, in the NodeRED approach, one should connect each single IOT Broker for each single sensor/device. While, in Snap4City case, a new block and service abstracting the whole set of devices is available and it has been called IOT Directory and made accessible also for NodeRED users. An IOT Application developer exploits the IOT Directory to search and discover the most suitable devices to be used, by searching through metadata and/or on map. Discovering capabilities based on the use of maps are offered that rely on the KB services.

According to Km4City solution [3], the city entities' models and relationships are stored into a KB grounded on Km4City ontology, while the historical and real-time data are collected in Apache Phoenix storage. In order to cope with the IOT devices and allow them to be discovered in the map, each device has to be registered on the KB with its attributes. While its current and historical values are accessible for data analytics and further reasoning. To this end, the IOT values have to be logged into the Hbase storage. This automated feeding process is performed by a set of processes (one for each IOT Broker) realized in Apache NIFI. The same set of processes also perform the indexing for general view, facet and timeline drill down on Developer Dashboard. In this manner, both historical/referral data and real-time data are accessible via Advanced Smart City API that exploit both the KB Services for spatial-temporal reasoning in SPARQL and Hbase/Phoenix in SQL, integrated. Moreover, the Advanced Smart City API are also accessible as MicroServices for IOT applications in NodeRED. To this end, to implement the MicroServices, a number of NodeRed Blocks have been realized and deployed into NodeRED tools. The Advanced Smart City API and Knowledge Services also provide data and information to Dashboards that can be produced by the newly developed Dashboard Builder tool.

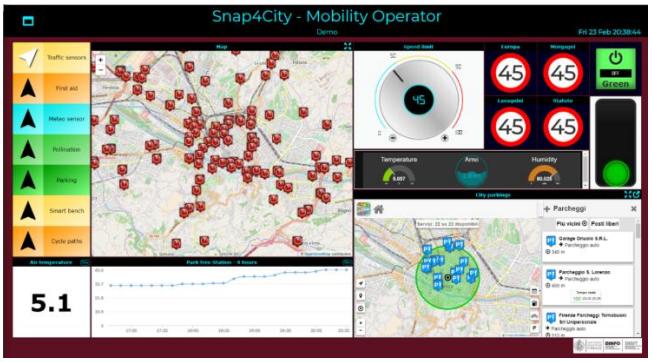
The Snap4City architecture is completed by a number of tools such as: Dashboard Builder for creating city dashboard and IOT actuator for them; ServiceMap for working with the KB and generating smart city API; a set of Developers Dashboards for monitoring consumed resources on the platform: data stored, cloud resources at level of VM (virtual machine), hosts and containers, and network traffic among MicroServices, IOT Applications, Mobile and Web Applications, and IOT/IOE devices, and ETL processes. As regarding the management of cloud resources for the IOT applications, ETL and Data Analytics processes are put in execution on a pool of VM as containers. Containers are realized by Docker while the Marathon-Mesos solution is used for their management. The whole set of VM constituting the solution is also allocated on cloud. Therefore, the system may keep under control the consumption of resources also controlling when they have to be turned on/off. Before passing to describe the performance analysis performed, a functional example is reported in the next section that can be regarded as a proof of a large number of the above-mentioned functional requirements.

## IV. An Example

In order to put in evidence the capability of the Snap4City solution with respect the IOT challenge presented by Select4Cities, and the state of the art tools, consider the following scenario of mobility and transport in which the following entities are involved:

- A Mobility Operator of the city or some mobility stakeholder in front of a web page which may be interested in understanding the city dashboard status in terms of real time and referral data regarding mobility and transport (traffic sensors, parking status and predictions, triage in the hospital, environmental data, etc.), and may act on the duration of the red-light semaphore in a specific case, and on speed limit in major paths of the city.
- One or more drivers sitting on their public vehicles and touring the city (such as a bus, an ambulance, a police car, a garbage collector track) that may need to monitor the city traffic conditions and environmental data, and may also need to act on the red-light semaphore (the one controlled by the Mobility Operator).

Each of them has his own Dashboard to control, one on the web and the other on mobile as depicted on Fig. 3. The Mobility Operator may change the speed limit acting on the dimer with double control (fine and by step), which is an IOT device encapsulated into the City Dashboard. Any action on IOT dimer provokes the sending of a message to an IOT Broker (IOT Orion broker). Thus, an IOT application has been designed to update (in data driven) the changes of the dimer (telemetry protocol) and thus to propagate the new speed limit to a number of Speed Limit Plates along a major path in the city. Also the involved Speed Limit Plates are IOT devices/actuators, which in turn are read by the former Dashboard to allow the Mobility Operator to verify if the information has been correctly propagated. In the same manner, the Speed Limit on the mobile of Driver is updated since he is registered on the limit corresponding to his location




(a) https://www.disit.org/dashboardSmartCity/view/index.php?iddasboard=MjU3


(b) https://iot-app.snap4city.org/nodered/nr15/ui

Fig. 3. (a) Mobility Operator, (b) the view on mobile device of the Driver

in that moment. The presented demonstrator can be accessed to play with by using the links in Fig. 3.

Both the Driver and the Mobility Operator may act on their corresponding Dashboard for obtaining the trend and real-time value of the measured traffic and parking conditions. The Mobility operator, may obtain that results by acting on the selector on the left side of the city dashboard, while the Driver acting on the map of the mobile IOT App (by using the finger keep pressed on the map for a while, equivalent to the right button of the mouse) and selecting the point, thus simulating the movements on the car.

In this integrated scenario, the Mobile application has been developed by an IOT/IOE application in NodeRED flow with the addition of Snap4City MicroServices as depicted in Fig. 4. In order to allow monitoring and assessing the network workload, in the flow, each data received/requested (RX) by the flow, and transmitted (TX) is also logged into the EventLog block. It collects the data, index them to monitor the flow by using the AMMA tool (Application and MicroService Monitor and Analyzer) for understanding the traffic on the infrastructure, implemented as a Banana Dashboard on a multi facet and sharded index on SOLR. A more elegant solution would be to hide the log into the Smart City MicroServices.

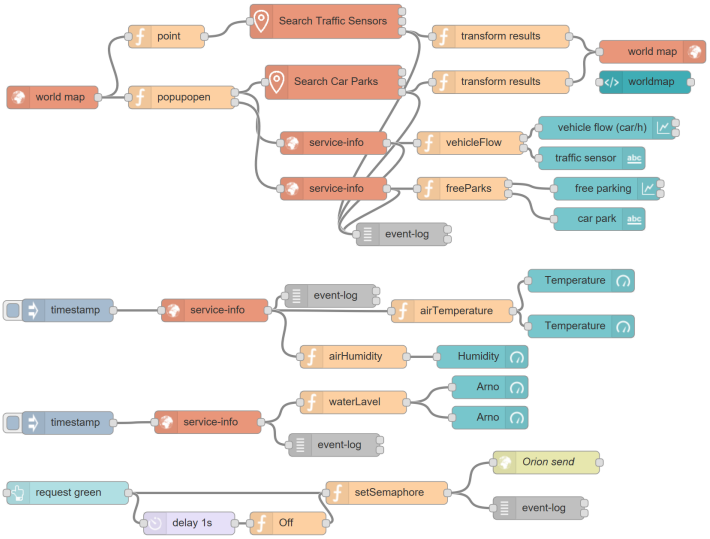



Fig. 4. IOT application in which data are collected from users actions, referral data from MicroServices of Smart City API Km4City and from IOT brokers.

In addition, the Mobile Operator may act on the red-light semaphore pressing the green button on the up-right corner of the Dashboard of Fig. 3. If the semaphore light is green, the duration of green is increased of 10s for each push. On the contrary, when the button is pressed, and it is red since 15s, it is changed to green, otherwise the change to green is delayed of 15s or red will be passed (this guarantees a minimum flow in the other direction). The same actions can be obtained by the Driver, who also sees the countdown until the semaphore changes its state. Other logic can be easily enforced (e.g. to make the reservation without acting on the red duration).

The above described logic and transformation has been implemented with the IOT application reported in Fig. 6. Also in this case, the monitoring of the traffic flow has been implemented by adding specific EventLog blocks. Please note that in the above example, (i) the Mobility Operator dashboard reported in Fig. 3 has been implemented by using the Dashboard Builder for Snap4City also including IOT devices embedded into the dashboard and exploiting real time data, referral data, and data driven events (received and produced), (ii) the IOT/IOE application in the hands of the Driver has been developed in NodeRED with the flow reported in Fig. 6, in which most of the functionalities have been realized through MicroServices developed by Snap4City enforcing access and exploitation of the Smart City API of Km4City, (iii) the IOT access has been realized using the services of IOT Directory and IOT Discovery tools for Snap4City, (iv) the EventLog has been used for monitoring the network usage by the IOT and ETL applications in the infrastructure. This latter issue will be more evident when in the following a view of the AMMA tool (Application and MicroService Monitor and Analyzer) will be discussed. The solution for IOT smart city has also to cope with security and privacy of the city users. To this end, the IOT-smart city solutions have to provide end-to-end secure

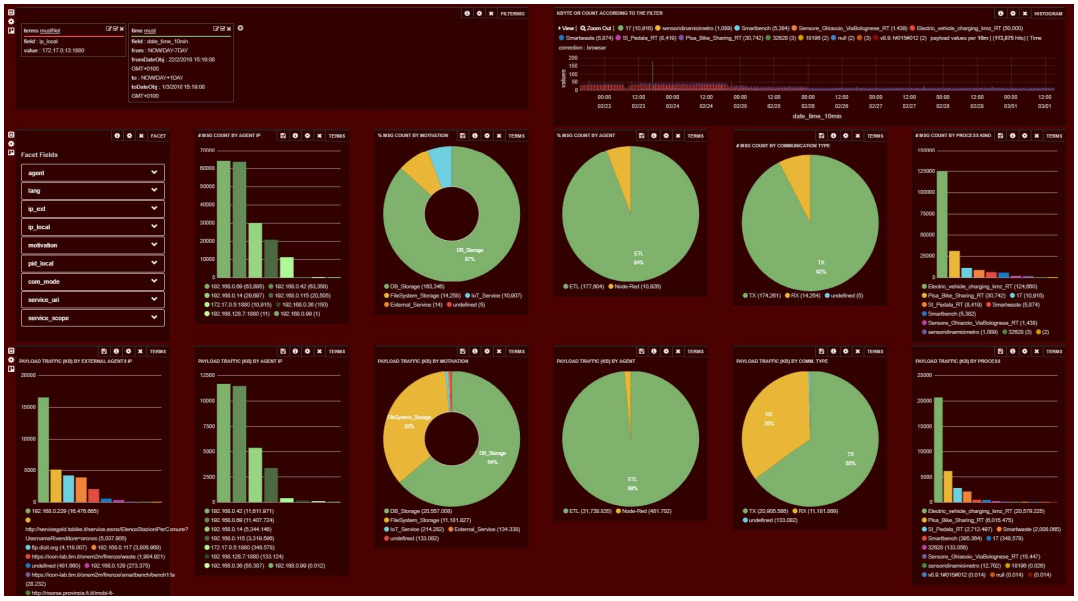


Fig. 5. AMMA tool: monitoring in real time data streams with facet search and interactive dashboard for segmenting the traffic: origin destination.

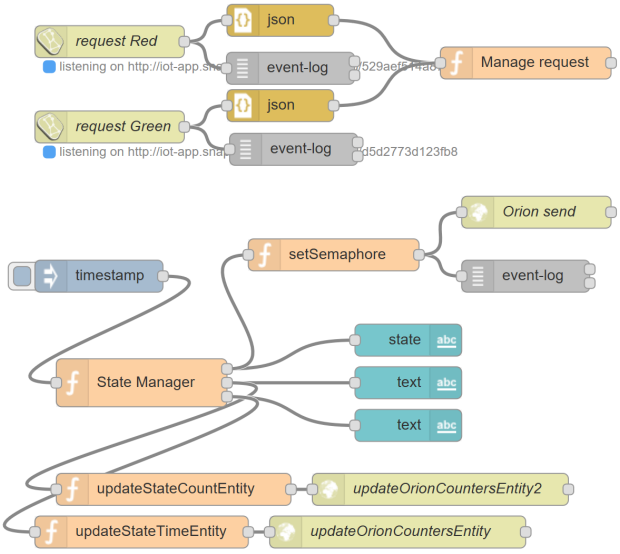



Fig. 6. IOT app. implementing the logic of red-light control of the semaphore

connection, for example using TLS/HTTPS messaging for the IOT and HTTPS for accessing the tools.

## V. Performance Analysis

The adoption of IOT applications in the context of smart city may imply to give at each single city user the possibility of having and programming one or more IOT Applications. These whole set of IOT applications and data processes running in the smart city back office have to be maintained under control, since some of them could be (i) intentionally developed for creating problems, (ii) developed by non-experts users that could produce IOT applications that in some unplanned manner may create large network traffic, (iii) adopted in terms of instances by many people thus creating a large number of instances performing the same actions, which will become computationally inefficient and will uselessly increase the costs. In the latter case, repeated functions/computations could be leveraged to a shared city service (IOT app as well or ETL), thus simplifying the costs and the flows. Therefore, the continuous analysis of the back-office network traffic and cloud workload helps the smart city operator to keep clean and sustainable the solution. Most of the IOT platforms have business models based on the number of messages or Kbyte exchanged. For these reasons, the performance analysis of the proposed solution has been conducted addressing two different aspects and scenarios:

A *Daily and real time*: the possibility of putting in the hands of developers and smart city managers an AMMA tool for real time monitoring and control of the consumption of communications bandwidth among the IOT applications, external services, storage, Micro-Services and smart city API, file system, etc.;

B *Sporadically for validation*: the assessment of the architecture in terms of maximum number of messages that can be managed by brokers and applications when a certain amount of cloud resources are available for running the processes that put in execution the IOT applications as containers on cloud.

### A. Assessing Communications Bandwidth

In order to assess the communication workload among the several services, MicroServices, processes in the smart city IOT back office, a large number of entities have to be monitored, while those that mainly provoke the traffic are the processes: ETL, DataAnalytics and IOT [3]. The communi-

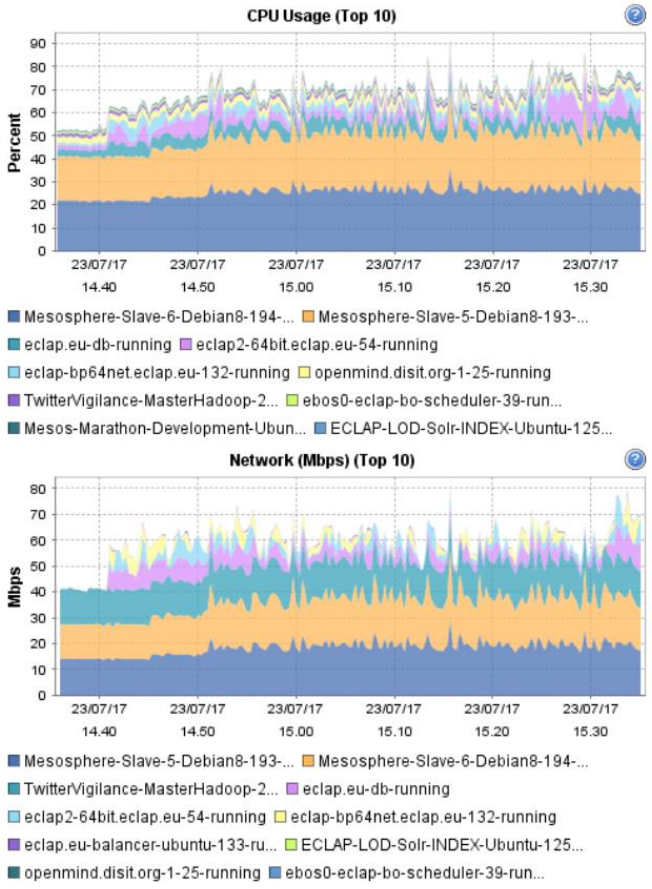



Fig. 7. Controlling messages sent / received per minute on the cluster of VM including container of IOT applications managed by Marathon-Mesos-Docker.

cations may be among applications, with respect to external services, due to periodic processes, accessing to the smart city API, etc. Therefore, the whole set of data flow volumes and messages have to be logged and the log data has to be easily partitioned and analyzed by the developers to understand over the time line and on map: (i) how much a given application, application area and services are communicating, in terms of volume and messages, (ii) how many messages are exchanged among applications, services, etc.[1] (iii) which are the top applications in terms of communication workload, etc. The communication activities are related to messages, for each of which one should have a clear view of its purpose and cost:

- transmitting/receiving data to/from;
- motivation: IOT (via IOT brokers or directly to devices), data store (save and load), dashboards (user interface interaction), file system access, external services via rest call API, WS, FTP, etc.; and Smart City API;
- protocol: MQTT, COAP, AMQP, HTTP, OneM2M, ..;
- process agent kind: ETL, Data Analytics or IOT apps;
- mastered by an IP-local to/from an external service at a given IP;
- and, in the case of external service, the URL should be a selection aspect as well as the service scope which can be internal or external to the smart city cloud.

For these reasons, an extension to the above-mentioned architecture has been realized to collect the stream of EventLog data and indexing them into a large SOLR index, thus allowing the access with a facet viewer to the whole set of data. And so, permitting at the developer to perform facet search and drill down over timeline as depicted in Fig. 5. Where, the first set of pie charts and histograms is representing the distribution of the volumes of message, while the second is representing the distribution of the flows in terms of message size. In the observed period, most of the messages have been due to the database storage of ETL in TX. While in terms of volume (message size), also the file system has been strongly involved and there is balance in TX/RX. From the latter histograms, it is also possible to identify which are the most exploited services involved: parking, smart benches, waste collector, bike sharing racks, electric vehicles.

By selecting with the mouse the single bars, it is possible to insert a filtering and request to all the other representation to recompute the graphs. In this manner, the analyst can study the contributions produced and received by each single IOT application towards all the other services, over time and kind.

### *B. Assessing Maximum Number of Messages*

In order to assess the maximum number of messages per minute which can be managed by a set of IOT applications with a certain number of resources, a test bed has to be set up. To this end, we have allocated 6 virtual machines, VM, on which 300 Docker Containers each of which with an IOT application processes having 196KByte of memory, for 79 GByte Ram. Each host had 24 cores at 2.299 GHz, on 128 GByte RAM: up to the 75% of resources on cloud, to keep the VM process in a safe/live range. Each single IOT Application was based on a NodeRED, which can sustain about (in that condition) 120 msg/s MQTT in input thus performing 120 REST Calls per second. Thus obtaining 240 events per second: 96.000 events per second per Host/Node.

In Fig. 7, a monitoring panel (also implemented as IOT application) for the workload is presented. On the left bottom corner the trend of the number of healthy IOT applications/containers is presented. So that, 300 healthy IOT applications have been registered. The test started by using a variable number of MQTT messages generated on a Mosquitto broker which was sending new values of data to the 300 allocated IOT applications. During the same test time, the amount of messages sent per second has been changed arriving at 1 Million of messages per minute. Each IOT application of the test, in effect, collects an MQTT message and sends a message to a MicroService of data store. Therefore, for the cluster, the total number of messages per minute was about 2 million. It has been reached by using a number of IOT MQTT brokers with the aim of arriving at the limit when

[1]Note that in most of the IOT applications on the market the prices exposed, and thus the business models strongly depend on the number of messages and/or the volume of data exchanged in terms of KByte/MByte.


Fig. 8. Monitoring cloud resources during the workload test for the IOT smart city applications

the maximum acceptable exploitation of memory and CPU resources was reached. The maximum workload has been fixed at the 75% of Cloud VM resource exploitation. When the limit is reached, the IOT Applications in Container are not capable to react, and thus the manager identifies them as not-running. Fig. 7 depicts the case which represents the limit for the resources available in that configuration.

The scalability of the solution may be obtained quite easily since the overhead of the solution is close to 13% for VM, ESX, Marathon per Node, plus an offset due to the needs of having 3 VM for the masters that may have managed up to 16 Hosts/nodes, thus doubling resource is needed every other 16 Hosts/nodes. In our test case, 6 VM have been managed. At the same time the monitoring of cloud resources has been obtained by using the cloud infrastructure tools as reported in Fig. 8. The picture represents a real condition in which VM with IOT contained managed by Mesos work together with other VM with other processes and services.

## VI. Conclusions and Future Work

Smart City solutions initially started with the production of open data and their collection on managing open data tools. Then smart cities evolved towards data aggregation and semantics modeling of city entities, benchmarks, and datasets. Recently, some of them are also offering IOT support. The combination of IOT and smart city is not an easy task, the data volumes are much higher than those addressed for industrial IOT. The complexity of IOT smart city solutions have been identified by a number of actors. The European commission started set up the EIP project for stimulating and concerting actions. On the other hand, the Select4Cities project of the European Commission and associated community `http://www.select4cities.eu/` created a challenge to find research-based solutions satisfying a formalized set of functional and nonfunctional requirements. Snap4City solution that has been presented in this paper, is one of the solutions approved and developed in response to that challenge. Snap4City solution is based on MicroServices directly provided by a number of applications and Smart City API. Snap4City users can develop in few steps sophisticated IOT applications exploiting MicroServices that can control city dashboards as well as IOT mobile applications. Moreover, a number of development and monitoring tools have been designed and developed. Among them, in this paper, a special attention is given to the tools and solutions for monitoring communication performance among IOT, storage, smart city API, external services, etc. Another aspect addressed in the paper consisted on the test to perform the assessment of scalability feature of the IOT smart city infrastructure in terms of maximum number of messages exchanged per minute fixed the number IOT processes and corresponding resources. Future work will be devoted to put together these aspects to develop algorithms for automated scaling and IOT cloud management.

## Acknowledgements

Our thanks goes to the Select4Cities Consortium for supporting our work with useful feedbacks about the real needs of smart cities like Antwerp, Copenhagen, and Helsinki. We also thanks to Km4City for the usage of data for testing and validation in the early phases.

## References

[1] Aberer K., Hauswirth M., Salehi A., "A middleware for fast and flexible sensor network deployment". In: Proc. 32nd Int'l Conf. on Very Large Data Bases, pp. 11991202, 2006.
[2] ArcGIS OpenData: `http://opendata.arcgis.com/`
[3] C. Badii, P. Bellini, D. Cenni, A. Difino, P. Nesi, M. Paolucci, Analysis and assessment of a knowledge based smart city architecture providing service APIs, Future Generation Computer Systems, Vol. 75, 2017.
[4] Bandyopadhyay S., Sengupta M., Maiti S., Dutta S., "Role of middleware for internet of things: A study", Int'l J. of Computer Science and Engineering Survey 2(3), 94105, 2011.
[5] Bellini P., Bruno I., Nesi P., Rauch N., "Graph Databases Methodology and Tool Supporting Index/Store Versioning", J. of Visual Languages and Computing, Elsevier, 2015.
[6] Calbimonte J.-P., Sarni S., Eberle J., Aberer K., XGSN: An Open-source Semantic Sensing Middleware for the Web of Things, In 7th Int'l Workshop on Semantic Sensor Networks, 2014.
[7] Compton M., Barnaghi P., Bermudez L., et al. "The SSN ontology of the W3C semantic sensor network incubator group", J. of Web Semantics 17, 2532, 2012.
[8] CitySDK: `http://www.citysdk.eu`
[9] CKAN: `http://ckan.org`
[10] Specification For Urban Platforms, EIP Project, version 2.2, 2016, European Innovation Partnership for Smart Cities & Communities.
[11] Green Button Connect: `http://www.greenbuttonconnect.com/`
[12] IoT Analytics. List of 640+ enterprise iot projects. `iot-analytics.com/product/list-of-640-iot-projects/`.
[13] J. Jin, J. Gubbi, S. Marusic and M. Palaniswami, "An Information Framework for Creating a Smart City Through Internet of Things," in IEEE Internet of Things Journal, (1):2-112-121, 2014. doi: 10.1109/JIOT.2013.2296516
[14] A. Krylovskiy, M. Jahn and E. Patti, "Designing a Smart City Internet of Things Platform with Microservice Architecture," 3rd Int'l Conf. on Future Internet of Things and Cloud, 2015, pp. 25-30. doi: 10.1109/Fi-Cloud.2015.55
[15] Lin F. J., Ren Y., Cerritos E., "A Feasibility Study on Developing IoT/M2M Applications over ETSI M2M Architecture." In Proc. of Int'l Conf. on Parallel and Distributed Systems, IEEE, 2013.
[16] Message Queue Telemetry Transport (MQTT), OASIS Std., 2014.
[17] S. Newman, Building Microservices. OReilly Media, Inc., 2015.
[18] `https://nodered.org/`
[19] OpenDataSoft: https://www.opendatasoft.com/
[20] Swetina J. et al. "Toward a standardized common M2M service layer platform: Introduction to oneM2M." IEEE Wireless Communications 21.3 (2014): 20-26.