A Scalable Multilayer Middleware for Distributed Monitoring and Complex Event Processing for Smart Cities

Alexandre Meslin
PUC-Rio
UFRJ
Rio de Janeiro, Brazil
meslin@inf.puc-rio.br

Noemi Rodriguez
PUC-Rio
Rio de Janeiro, Brazil
noemi@inf.puc-rio.br

Markus Endler
PUC-Rio
Rio de Janeiro, Brazil
endler@inf.puc-rio.br

***Abstract*—In this paper, we present a hierarchical, distributed, cloud-based, context-aware architecture for collecting, processing and distributing data in a smart city. The architecture we are proposing has three hierarchical levels, and supports complex event processing (CEP) in several of them. At the lower layer, several mobile objects work as interfaces to sensors and actuators and provide processing capability for local filtering and detection. A second level, consisting of gateways and processing nodes, processes information from its locale and sends the data received from the mobile objects to the storage level using the existing network infrastructure. The highest level provides support for structured storage and queries of the information. Applications outside the platform can collect data through the external interface of the highest level. The system is under development using the InterSCity platform in the upper layer, the ContextNet middleware at the middle layer, and the Mobile-Hub application at the lower layer. It currently collects and processes data on buses running in the City of Rio de Janeiro.**

***Index Terms*—Smart City; Internet of Things; Fog Computing**

I. Introduction

The use of sensors and actuators is increasingly present in several cities around the world. Temperature, pressure, humidity, air quality, brightness, vehicle movement and pedestrian presence sensors help the governance of smart cities. Actuators, such as public lighting poles, traffic lights and alarms, help administrators to control events in a large city efficiently.

All these sensors and actuators generate a very large volume of data and a high demand for processing that needs to be organized. For example, in 2016, the City of Barcelona had more than 1,800 sensors that generated 1,300,000 data per day [1]. One of the greatest challenges inherent in the design of an smart city is the management of information, from the moment it is captured to its final use. Separating urgent data according to its local or global relevance allows a better utilization of the bandwidth and at the same time decreases the latency, improving system scalability.

This volume of data refers to the city of Barcelona, which has a population of approximately 1,600,000 inhabitants living in an area of just over 100 km^2. There are several cities in the world that are much larger than Barcelona, both in terms of area and population. Considering that the number of sensors in Barcelona should increase, consequently increasing the volume of data generated by them, and if we extrapolate that amount to the larger cities, we realize that the volume of data to be transmitted and processed will be huge. If the system that receives and processes this data is centralized, there is a great chance of creating a bottleneck due to system or network overload. This bottleneck can lead to problems such as data loss and/or delayed responses. A hierarchical system able to filter data and process information locally will relieve the higher levels and will be able to provide faster response to events.

There are currently several smart city initiatives, where sensors and actuators provide data and respond to stimuli to facilitate the management of day-to-day activities in the cities. According to the Forbes website[1], Barcelona, New York, London, Nice and Singapore are the top five "smartest" cities. A smart city should efficiently manage energy consumption (Smart Grid), vehicle traffic and parking areas, street lighting, etc. through integration between the various government agencies, companies and the population, providing open data to all of them.

In this work we present MUSANet[2] (Mobile Urban Sensoring and Actuation Network), a distributed hierarchical context-aware system architecture for capturing, storing and processing sensor data, sending data to actuators, and receiving and publishing information through publish-subscribe protocols. MUSANet is based on the InterSCity [2] platform and the ContextNet middleware [3]. Data is captured using the Mobile-Hub [4] application. Events are analyzed in real time through CEP[3] - Complex Event Processing [5].

Our work has an intersection with Participatory Sensing [6] by using smartphones or smart mobile devices to collect

978-1-5386-5959-5/18/$31.00 ©2018 IEEE

[1] https://www.forbes.com/sites/peterhigh/2015/03/09/the-top-five-smart-cities-in-the-world/#750c956c67ee

[2] Available at https://github.com/meslin8752/InterSCity-onibus

[3] We use the engine from Espertech (http://www.espertech.com)

and analyze sensor data before sending them to the core of the infrastructure in the cloud. This pre-processing allows to decrease the amount of data that is transfered over the network while allowing faster responses to local events.

The rest of the paper is organized as follows. In Section II we describe the technologies used. We describe the architecture in the Section III and we present an example of a scenario in Section IV. Related works are presented in Section V. Section VI concludes the paper and presents next steps for future researches.

II. Enabling Technologies

This section presents the main technologies used in the implementation of the proposed architecture.

A. InterSCity

The InterSCity [2] open-source[4] microservice platform is designed to collect, store in a structured way, and to manage sensor information. InterSCity provides the basic blocks for the development of applications related to smart cities through REST APIs.

Figure 1 presents the platform's architecture. IoT gateways can register new features, send sensor data or sign up to receive actuator commands. The Resource Catalog module stores information about the city's static features, while the Data Collector stores sensor data, allowing external programs to access real-time or historical data stored. Actuation commands are sent from external applications to the Actuator Controller module which sends the command to actuators subscribed to the platform. Search for information is facilitated by the Resource Discovery module, which allows the data be selected by location, values ranges, etc. The Resource Viewer module acts as a front-end service, allowing the creation of external applications that visually explore the city's resources.

B. ContextNet

The ContextNet[5] middleware allows context to be added to information from sensors in large-scale applications. Its implementation is based on SDDL using the OMG DDS protocol [7] at its core, and MR-UDP [8] for bidirectional connection with mobile devices, as shown in Figure 2. The three main components of the ContextNet interface are (1) Gateways, responsible for providing a connection point for mobile nodes or smart objects; (2) PoA-Managers (Point of Access Manager), which manage the list of gateways, periodically informing mobile nodes the best (nearest) gateway according to the node position, while allowing load balance; and (3) GroupDefiners, which define the group to which each mobile object connected to ContextNet belongs to, based on its position. The SDDL protocol allows a large number of Gateways besides general purpose Processing Nodes.

The DDS protocol was developed to work using multicast or unicast communication over UDP/IP, which restricts its application to local networks. In section III we will discuss how to implement ContextNet as a fog application using the Internet infrastructure.

[4]Available at https://gitlab.com/smart-city-software-platform/dev-env

[5]Available at http://www.lac.inf.puc-rio.br/dokuwiki/doku.php

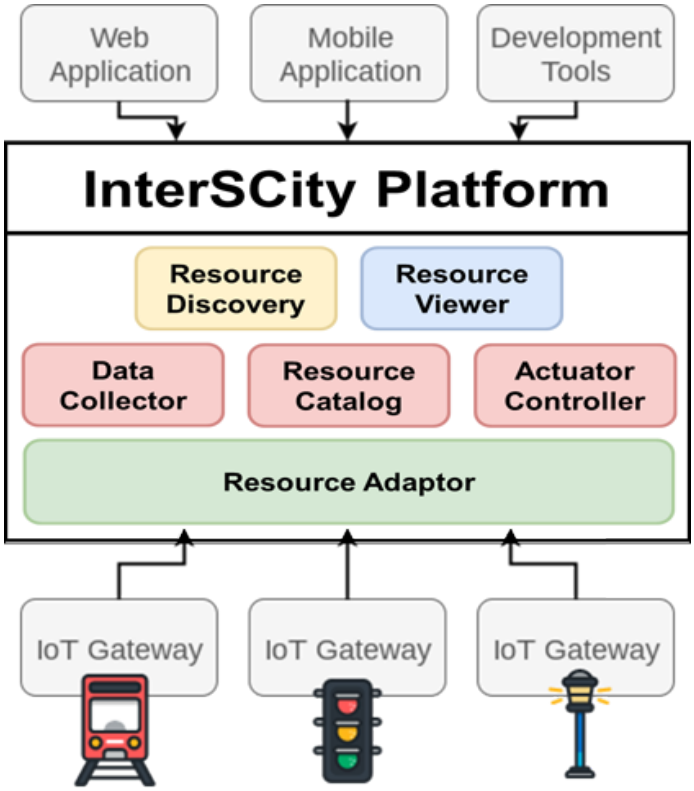

Fig. 1. InterSCity Platform Architecture [2].

C. Mobile-Hub

Mobile-Hub[6] [9] is an application for Android devices that provides connectivity to mobile devices using Bluetooth technology. Designed to be an extension of the ContextNet, it acts as a bridge between sensors and the Internet, adding local information processing capability with support for complex event handling (CEP) [5]. A new version under development will also include a local database which can store a small amount of data in the mobile device.

III. The MUSANet Architecture

In an smart city, sensors produce a myriad of information. Some applications need this information in real time. Street lighting control and traffic light synchronization are examples of this type of applications. Other applications need to analyze historical data to generate reports or to serve as a base for city planning. Typically climatic or seasonal events need to be stored and compared by these applications. Traffic information falls into these two categories: it needs to be processed in real time so that traffic lights timing is properly configured, but it also needs to be stored so that city hall can make long-term decisions such as those regarding the development of the city's road network.

[6]Available at http://www.lac.inf.puc-rio.br/dokuwiki/doku.php?id=m_hub

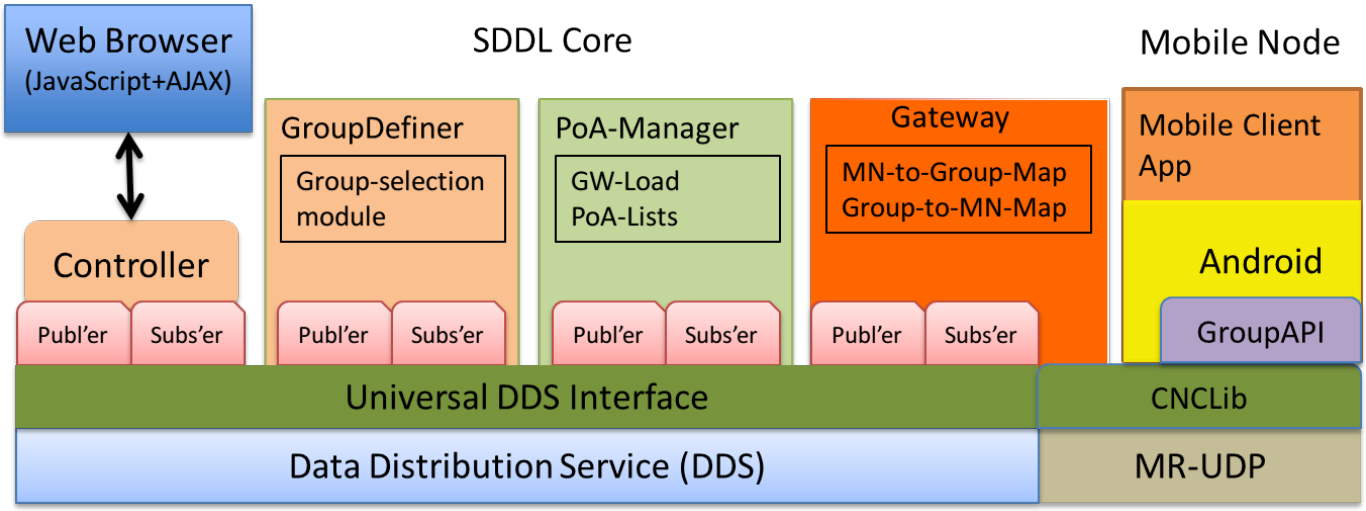

Fig. 2. ContextNet Architecture [8].

Although architectures such as InterSCity are intended to be extensible and scalable, other elements inherent in communication, such as the network between the sensor and the platform, create bottlenecks and delays in the system. To deal with a geographic scale, we propose a scalable hierarchical architecture that reduces response time by implementing a fog distributed system where BLE[7]-enabled smart nodes running the Mobile-Hub-based application collect data from nearby sensors and perform local processing. The resulting data are sent to distributed access points so that there is always a gateway to the ContextNet close to the mobile object. The ContextNet middleware will be responsible for contextualizing the data and forwarding it to the nearest InterSCity platform, according to the geographical location of the data source. In the local network of this gateway there will be a Processing Node and an InterSCity platform, allowing information to be processed and/or stored quickly. Actuators registered in ContextNet receive commands relative to their regions, avoiding floods of unnecessary commands to unrelated actuators.

MUSANet has three layers reflecting different levels of abstraction as shown in Figure 3. InterSCity implements the top layer, forming a structured storage ecosystem, supporting high-level queries. The central layer, implemented by ContextNet middleware, allows the allocation of sensors and actuators into groups or regions, and provides distributed access points to the infrastructure. The lower layer consists of smart mobile devices running Mobile-Hub to connect Bluetooth-enabled sensors and actuators to the MUSANet infrastructure through Wi-Fi or 3G/4G network.

The top layer of MUSANet provides structured information processing and the storage module. This module also provides an API for developing applications for end users or other systems that need the stored data. The middle layer is responsible for receiving data from mobile or fixed nodes connected to

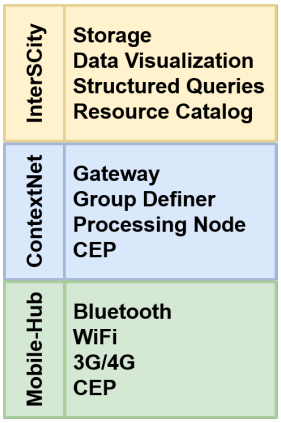

Fig. 3. MUSANet 3-Layer Architecture.

sensors. This layer is also responsible for defining the groups (regions) to which the mobile nodes belong based on their geographic position. Also at this level, complex events formed from data from one or more sensors scattered throughout a region can be analyzed in real time allowing rapid response. At the lower layer are mobile and fixed devices that may have the role of sensors and/or actuators, but can also work as bridges between BLE-enabled sensors and actuators. This layer can also analyze complex events based on some local sensors, that is, connected to the device. The result of this analysis can be sent to the upper layer and/or generate a reaction through local actuators.

MUSANet uses various distributed ContextNet sites that are connected through IP tunnels using the Internet infrastructure to create one ContextNet infrastructure. Several network topologies can be used, including star, hierarchical or full-mesh format, with or without path redundancy. The topology, implementation, and architecture of the IP tunnels are not within the scope of this article and will not be covered in

[7]BLE - Bluetooth Low Energy

detail.

In our architecture, the smart city is divided into groups or regions based on sensor distribution and not just neighborhood or zones. These regions can (and should) have intersections, and there is also the possibility that regions encompass entirely more than one region, as exemplified by Figure 4, where region B, c and D have intersections with two and three regions while region A is completely disjoin. Although not shown in the Figure, there may be regions covering completely other regions.

For example, to implement the regions in MUSANet for public transportation in a city such as Rio de Janeiro, an administrator could be based in the neighborhoods, where clusters of adjacent small neighborhoods interconnected by a large set of bus lines could compose a single region, and larger neighborhoods could be divided into more than one region. Each bus stop could be assigned to a micro-region completely within the region defined by the neighborhood, as previously described. Very close bus stops along the same route could belong to the same region.

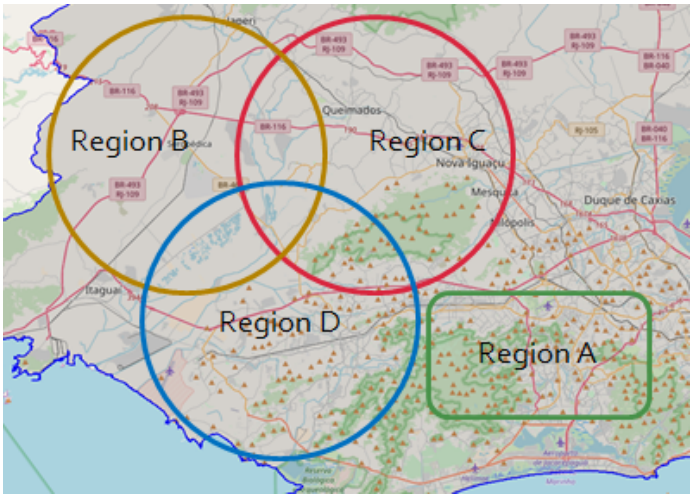

Fig. 4. Example of regions in a city.

Each region must have a set consisting of at least one Gateway, a GroupDefiner, a Processing Node, and an InterSCity instance connected through a local area network. Local networks are connected over the Internet structure using IP tunnels, forming a single local area network. The implementation of these elements can be done on a single or on multiple machines. Figure 5 shows the block diagram of our architecture with four groups of Gateways, InterSCitys, and Processing Nodes. The MUSANet infrastructure needs at least one PoA-Manager (not shown at Figure) to send the nearest MUSANet Gateway address to the mobile nodes.

Multiple instances of Mobile-Hub installed on mobile objects can collect sensor data, for example, using BLE. The collected data can be analyzed locally through CEP rules configured on the device or sent via Wi-Fi or mobile network to the infrastructure using the nearest ContextNet Gateway.

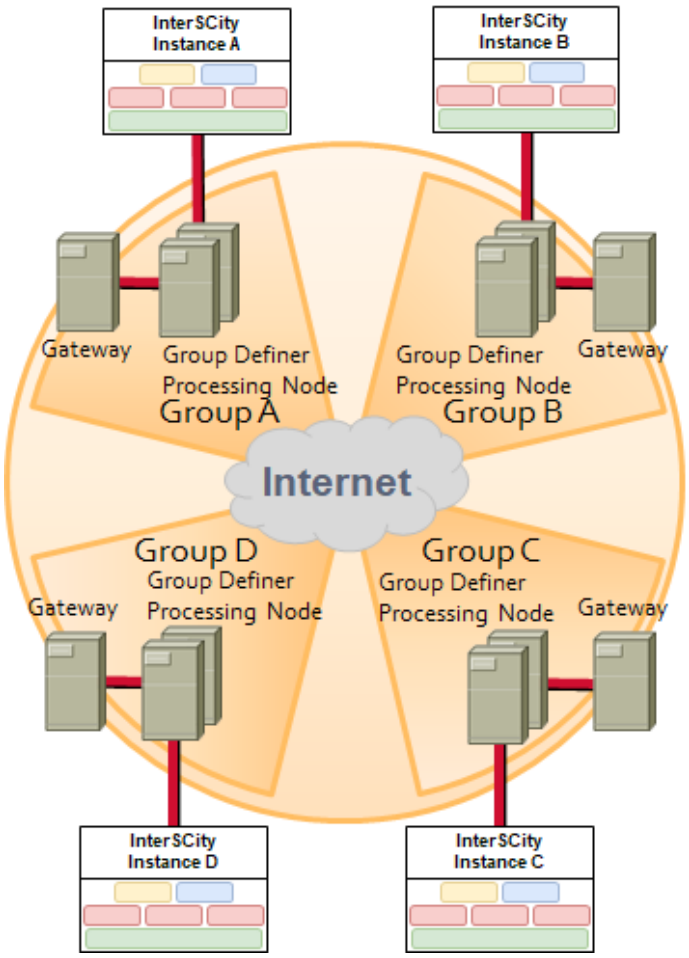

Fig. 5. MUSANet diagram.

Using ContextNet to add region context to data allows us to distribute the data flow across multiple gateways, increasing network throughput while decreasing latency. Mobile objects should send their location to the GroupDefiner to find out which groups they belong to. From the group information, the mobile object can locate its nearest gateway to ContextNet. Future messages will be sent to the Processing Node of this group through the new gateway until the object moves to the region of a new group with a new gateway. If the object is at the intersection of two or more groups, the object must send the information to the Processing Nodes of all groups where it is located, using its respective gateways.

Each Processing Node is programmed to process only certain types of data. Data sent by sensors that are not to be processed by the Node can be discarded since another node responsible for processing will also receive the same data. As the data arrives at the Processing Node, the CEP engine scans the data in real time and triggers actions, when applicable. Depending on the data type, the Processing Node can send it to be processed and stored in the InterSCity located on its local network. The data are sent from the Processing Nodes to

the InterSCity via HTTP[8]. Sensors and actuators that do not register with ContextNet, regardless of the reason, can directly contact any instance of InterSCity via HTTP protocol.

The MUSANet architecture is not dependent on the technologies selected in the Section II. Other technologies can be used at each level of the hierarchy. For example, the Dimmer [10] platform which is a microservice-based platform that enables device and capability discovery, and provide an API for real-time and historical data access, can be used to manage the data base.

Another example is the FIWARE [11] platform, an OpenStack and Docker based environment that stores data hierarchically in a MongoDB database, supporting both sensors and actuators. The platform also provides support for users to be authenticated and users roles can be defined with configurable permissions. Unlike InterSCity, the platform natively supports complex event analysis through CEP for real-time event handling.

The Smart Streets IoT Hub described in [12] and the gateway presented by Aloi et al. in [13] can be considered two alternatives for using the Mobile-Hub. The first was developed to interface with the Hyper/Cat platform [14]. The second is a smartphone gateway with support for Wi-Fi, Bluetooth, Lte (H+, 3G, etc.) networks and even ZigBee via a Micro-SD card from Spectec[9].

IV. Exploring the Proposed Architecture

Although there are several simulators and models of smart object movement in cities [15] [16] [17], in order to perform more realistic tests we chose to use data from the municipal buses made available by the City Hall of Rio de Janeiro[10]. This data is published approximately every 30 seconds and provides the position in geographic coordinates, speed, line number, serial number, and date and time of transmission of the information for each bus[11]. Data from temperature, pressure and light sensors were generated by us to simulate the existence of sensors that are likely to be available in the near future.

Currently, buses in Rio de Janeiro City are not running the Mobile-Hub application, so, we created a program to read data from the City Hall website and send it to ContextNet, acting as a bus running Mobile-Hub. This program obtains the data in a JSON structure that has information of all the buses in operation. For each bus, the program prepares an InterSCity-compliant JSON string and sends it to the ContextNet Gateway, theoretically, closer to the bus to simulate the expected behavior of a Mobile-Hub-equipped bus. The ContextNet gateway sends the data to the Processing Node for processing and storage in its InterSCity instance.

[8]By default, elements within the ContextNet network use DDS protocol, but to avoid unnecessary modifications to the Platform, InterSCity does not need to register within ContextNet and continues being accessed through its native protocol.

[9]http://www.spectec.com.tw/sdz-530.html

[10]http://dadosabertos.rio/dataset/onibus-gps-2

[11]To increase volume of data, we generate artificial information through data interpolation.

With this architecture configuration, we can perform some experiments as described below. Only the operation of the Mobile-Hub cannot be fully tested because we do not have the application installed in the objects that will be collecting data. During testing, the Mobile-Hub will function only as an actuator, receiving data from the platform.

To perform the first integration tests of the various components, we developed an application using InterSCity, ContextNet, CEP, and Mobile-Hub. The application consists of the following parts: (1) a module to capture and to generate the data, performed as described above; (2) another module with a Gateway, a GroupDefiner and a Processing Node which receive information from buses and send them to be stored in the appropriate InterSCity, according to the group where the bus is located; (3) InterSCity, which manage the information storage, while making it available for queries; and (4) a user interface, an Android application that allows the user to visualize the location of each bus that is in one of their regions (groups).

The following are some applications that take advantage of this integration. These applications make greater use of the distributed and hierarchical resources of the proposed architecture, in addition to making more intensive use of CEP analyzes.

Arrival time interval between buses of the same line at each bus stop: this application is an extension of the application described earlier, but using all levels of the architecture hierarchy. In this application, we consider that the Mobile-Hub is installed not only on buses but also on the smartphones of passengers and drivers. In order to use the application, the user must allow their cell phone to register in the ContextNet through the Mobile-Hub. When the bus leaves a bus stop, a timer of a CEP rule installed on the cell phone is trigged as shown in Figure 6. Considering that the amount of the time that a bus needs to go from one bus stop to next one has already been cataloged, if the timer exceeds the configured value for the bus to arrive at the next bus stop, the user will be informed about the delay.

```
select count(*)
from BusStop.win:
time(MAXTIME1 min)
having count(*) = 0
```

Fig. 6. EsperTech CEP code to detect bus delay running at a passenger smartphone.

At the hierarchical level immediately above, through CEP rules, as shown in Figure 7, configured in the Processing Nodes of each ContextNet block, based on the event `BusEvent`,which occurs every time a bus arrives at a bus stop, we can trigger alarms (`BusLateEvent`) whenever the time between two buses from the same line arriving at a bus stop is longer than the expected time. These alarms may indicate a transient problem, such as an accident that is causing an unexpected bottleneck. In addition to being sent to an operator for real-time verification, these alarms can be stored

at a InterSCity instance for historical analysis. The alarm `BusLateEvent` can feed another CEP rule to trigger another alarm if the number of delayed buses exceeds the limit as shown in Figure 8. In this case, this second alarm can also be sent to an operator to support real-time decision making as well as be stored in InterSCity so that a subsequent audit on bus quantity, line path or even the intervals can be made.

```
select count(*, line=BUSLINE)
from BusEvent(line=BUSLINE).win:
time(MAXTIME2 min)
having count(*) = 0
```

Fig. 7. EsperTech CEP code to detect bus delay running at a ContextNet Processing Node.

```
select count(*)
from BusLateEvent.win:
time(MAXTIME3 min)
having count(*) > MAXOCCURS and
BusLateEvent.line = line
```

Fig. 8. EsperTech CEP code to detect consecutive delays running at a ContextNet Processing Node.

At the top of the hierarchy, late- and delayed-bus alarms are stored and can be audited to verify the quality of the service provided. The InterSCity can also be used in this scenario to warn bus users who are logged into ContextNet as described earlier.

Traffic Jam warnings: bus users can be informed in real time about problems related to traffic jams or unexpected delays. Through their mobile running an instance of MobileHub, the user will be registered in ContextNet and will belong to at least a group in a region. If the system detects an anomaly in the bus flow, a warning directed to all users who are near the bus stops in the area can inform them of the situation so that they can look for alternative locomotion. To optimize the sending of messages, each bus stop or near set of bus stops must belong to a region that physically encompasses only the surroundings of the points. Note that these regions will be within other larger regions and must be as small as possible, but large sufficient to capture all nearby buses. A large region may consist of a few blocks. Figure 9 shows eight bus stops inside its microregions and all of them inside a larger region.

V. Related Works

In this Section, we present some Smart Cities implementations and discuss how they compare to our architecture.

The Barcelona City [18], in Spain is an example of success in the implementation of a smart system. Different types of sensors and actuators are spread throughout the city. Data is sent over the WAN to the platform via the Sentilo module[12], an open source application. The data collected by Sentilo is filtered, processed and stored in an intermediate layer that makes it available for applications that provide services to the city. The Barcelona system performs information processing in the central layer and in the application layer.

The SmartSantander [19] project consists of a testbed developed in Santander, in Spain. Several types of sensors were scattered around the city creating the largest testbed in the world. The architecture of the project is divided into three parts: (1) IoT nodes, composed of several IoT device, responsible for capturing the sensor information; (2) gateways, which perform the connection between the IoT nodes and the central infrastructure, and also serve as a bridge between the IEEE 802.15.4 networks and the Internet; and (3) the layer formed by the servers, which store, process, and deliver data to external applications. Although the gateway layer is distributed, the purpose of the distribution is not to provide multiple access points in order to divide data traffic, but simply to connect networks that do not use TCP/IP to the infrastructure.

The University of Padova, Italy, together with the city hall of Padova and Patavina Technologies, developed a smart city project [20] able to monitor public lighting and atmospheric conditions such as temperature and humidity. In this architecture, the sensor nodes installed on the poles create a WSN based on the IEEE 802.15.4, 6LoPAN and IPv6 RPL protocols. Gateways scattered around the city work with a dual stack protocols converting the sensor network protocols to Ethernet, IPv4, allowing data to be sent to the database through the standard network infrastructure.

ClouT [21] is a hierarchical three-tiered platform. The middleware was used in implementations of the smart cities of Santander, Fujisawa, Mitaka and Genova. Its lower layer is able to receive sensor data and to send commands to WSAN actuators. The top layer allows access to the collected data. A third layer addresses security and dependability issues on the platform. Although the lower layer had processing capacity, data is sent to the processing and storage unit without any prior processing, while in MUSANet events can be processed at any level.

Khan et al. [22] implemented a three-layer infrastructure similar to MUSANet. The main point of the work presented is based on services to citizens focused on quality of life. In their implementation, the lowest layer of data acquisition is distributed in a cloud rather than over the fog. According to the article, the IoT layer is not part of the architecture proposed by the author, but it only provides information for the platform.

In our architecture, we distribute computing into three hierarchical levels. Each level is able to perform information processing as follows: (1) at the point of collection, next to the sensor; (2) at the interface between the sensors and the system core; (3) and/or in the system core. The lowest level, where sensor data is collected and commands are sent to the actuators, is implemented in a fog, allowing information to be quickly delivered to and processed by MUSANet, differently from how it is implemented in Barcelona and Santander, where the gateways work as simple bridges between the

[12]http://www.sentilo.io/wordpress/

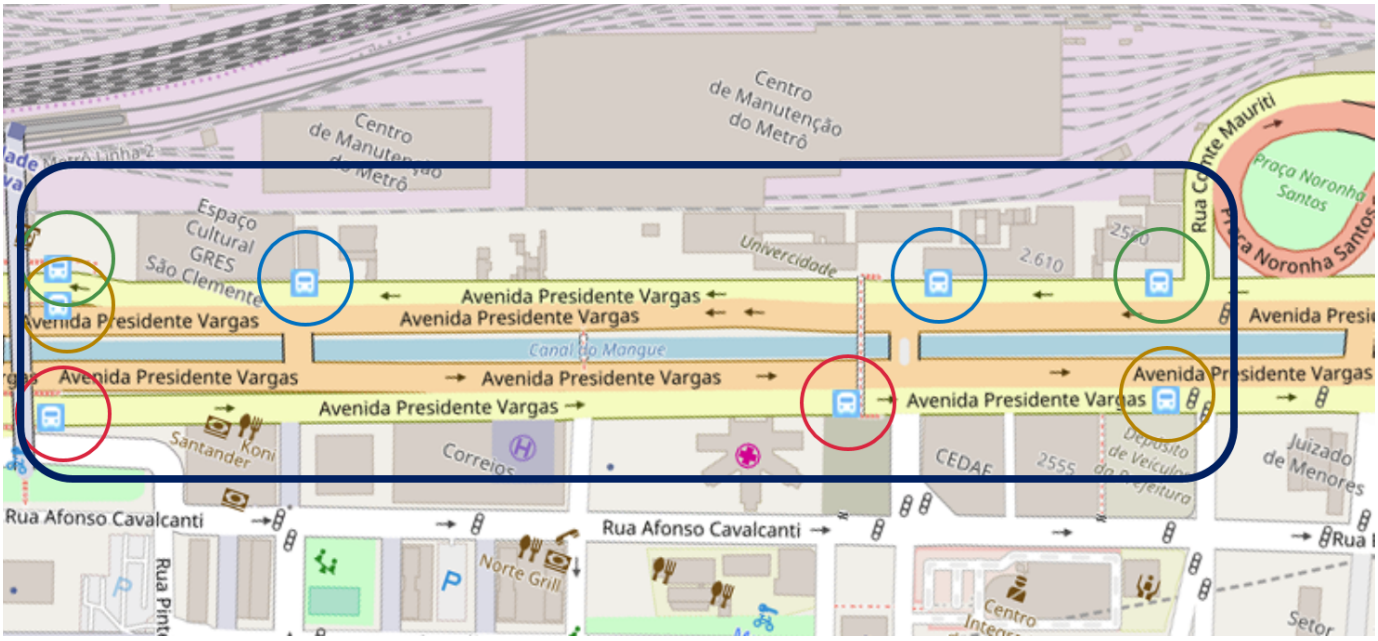

Fig. 9. Bus stops and their microregions within a larger region.

WSN protocols and TCP/IP. Unlike the architectures presented previously, in MUSANet, using the Mobile-Hub and the data processing capacity of its host mobile device, the sensors are not just passive agents or simple data generators but become units able to perform some local pre-processing of collected data. Consolidated information from the mobile devices is transmitted through the mobile network infrastructure to the various ContextNet access points (gateways). The top two hierarchical levels are implemented in a cloud, taking advantage of the existing Internet structure.

VI. Final Considerations

The proposed architecture presented in this article allows data from sensors to be processed at several hierarchical levels in a distributed way in order to minimize data traffic and its delay. Information can be analyzed in real time or stored for historical analysis. Data processing can be used to trigger alarms which can be stored and/or used to feed other alarms. Alarms and applications outside the proposed architecture can be used to send commands to actuators in specific regions. The ContextNet is able to provide multiple Gateways to MUSANet infrastructure that can be spread throughout the city. The PoA-Manager will ensure that the mobile object will always be aware of the best ContextNet Gateway to connect to. High-scale performance will be analyzed in a future work.

The SmartSantander project distributed approximately 1,500 sensors [23] in Santander city, Spain, with 35 km^2 and 178,465 inhabitants. Barcelona city has more than 2,000 sensors [1] spread over an area of 100 km^2 and 1.6 million inhabitants. A project equivalent to that of Barcelona, in the city of Rio de Janeiro, Brazil, which has a population of 7.5 million inhabitants and an area of 1,580 km^2, would need from 10,000 to 30,000 sensors. To support such a large number of sensors, the MUSANet project must be scalable. Preliminary testing has shown that each ContextNet Gateway supports more than 4,000 connected mobile devices simultaneously.

Other applications that can use MUSANet are:

Air quality: By analyzing complex events on the Mobile-Hub and ContextNet, alarms can be triggered whenever the concentration of pollutants exceeds reasonable limits. Monitoring the air quality can be performed continuously so that pollutants are detected immediately. Alternatively, CEP rules can also be used to verify that these elements remain at a very high concentration for a longer period, indicating that either the pollutant remains active or the weather conditions are not favoring the dissipation of pollution in the region. Both alarms described above can be used to feed another event analysis that will trigger a new alarm if the number of times critical air conditions exceed a reasonable limits. All alarms will be stored in the InterSCity so the historic information can be analyzed later.

Street lighting control: Light sensors can be used to monitor street lighting. Decrease or lack of brightness can indicate faults such as burned out light bulbs, lack of energy or even that vegetation or some other element is covering up the bulbs. The information is stored in the InterSCity and analyzed in real time through CEP rules installed in ContextNet so that an action can be performed in real time.

Weather forecast and catastrophe forecast: Through sensors to detect the presence of garbage in culvert, atmospheric pressure and temperature sensors and humidity of the air sensors installed along the radio range of the bus routes, the responsible city hall departments can monitor the atmospheric conditions in each point of the city. In the eminence or presence of heavy rains, with clogged culverts or with much trash

in the region, administrators can proactively take appropriate actions to avoid further problems to the inhabitants, such as informing them through the Mobile-Hub, cleaning the culverts involved, or even divert the traffic to less affected streets.

We hope to add soon other instances of InterSCity and more Gateways and Processing Nodes to the proposed architecture for more complex testing.

ACKNOWLEDGEMENT

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq, proc. 465446/2014-0, CAPES, proc. 88887.136422/2017-00, and FAPESP, proc. 2014/50937-1 and 2015/24485-9.

We thank Arthur Del Esposte, Tallys Martins, and all IME/USP team for the support during the installation and configuration of InterSCity.

REFERENCES

[1] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, E. Marín-Tordera, J. Cirera, G. Grau, and F. Casaus, "Estimating Smart City sensors data generation," in *Ad Hoc Networking Workshop (Med-Hoc-Net), 2016 Mediterranean*. IEEE, 2016, pp. 1–8.

[2] D. M. Batista, A. Goldman, R. Hirata, F. Kon, F. M. Costa, and M. Endler, "InterSCity: Addressing Future Internet Research Challenges for Smart Cities," in *Network of the Future (NOF), 2016 7th International Conference on the*. IEEE, 2016, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7810114/

[3] M. Endler, G. Baptista, L. Silva, R. Vasconcelos, M. Malcher, V. Pantoja, V. Pinheiro, and J. Viterbo, "ContextNet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking," in *Proceedings of the Workshop on Posters and Demos Track*. ACM, 2011, p. 2.

[4] B. d. T. P. Gomes, L. C. M. Muniz, F. J. da Silva e Silva, D. V. dos Santos, R. F. Lopes, L. R. Coutinho, F. O. Carvalho, and M. Endler, "A Middleware with Comprehensive Quality of Context Support for the Internet of Things Applications," *Sensors*, vol. 17, no. 12, p. 2853, 2017.

[5] D. Luckham, "The power of events: An introduction to complex event processing in distributed enterprise systems," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*. Springer, 2008, pp. 3–3.

[6] F. Restuccia, S. K. Das, and J. Payton, "Incentive Mechanisms for Participatory Sensing: Survey and Research Challenges," *CoRR*, vol. abs/1502.07687, 2015. [Online]. Available: http://arxiv.org/abs/1502.07687

[7] OMG, "The Real-time Publish-Subscribe Protocol (RTPS) DDS Interoperability Wire Protocol Specification Version 22." [Online]. Available: http://www.omg.org/spec/DDSI-RTPS/2.2

[8] L. Silva, M. Endler, and M. Roriz, "MR-UDP: Yet another reliable user datagram protocol, now for mobile nodes," *Monografias em Ciência da Computação, nr*, vol. 1200, pp. 06–13, 2013.

[9] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. d. S. e Silva, "The mobile hub concept: Enabling applications for the internet of mobile things," in *Pervasive computing and communication workshops (PerCom workshops), 2015 IEEE international conference on*. IEEE, 2015, pp. 123–128.

[10] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a Smart City Internet of Things Platform with Microservice Architecture." IEEE, Aug. 2015, pp. 25–30. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7300793

[11] FIWARE Foundation e.V., "FIWARE | Open Source Platform for the Smart Digital Future," 2018. [Online]. Available: https://www.fiware.org/

[12] M. Blackstock and R. Lea, "IoT interoperability: A hub-based approach." IEEE, Oct. 2014, pp. 79–84. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7030119

[13] G. Aloi, G. Caliciuri, G. Fortino, R. Gravina, P. Pace, W. Russo, and C. Savaglio, "Enabling IoT interoperability through opportunistic smartphone-based mobile gateways," *Journal of Network and Computer Applications*, vol. 81, pp. 74–84, Mar. 2017. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1084804516302405

[14] R. Lea, "Hypercat: an iot interoperability specification," Lancaster University, Book/Report/Proceedings 69124, Apr. 2014.

[15] PTV Planung Transport Verkehr AG, "Traffic simulation with PTV Vissim for efficient junction design," 2001. [Online]. Available: http://vision-traffic.ptvgroup.com/en-uk/products/ptv-vissim/

[16] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *Journal de physique I*, vol. 2, no. 12, pp. 2221–2229, 1992.

[17] O. Biham, A. A. Middleton, and D. Levine, "Self-organization and a dynamical transition in traffic-flow models," *Physical Review A*, vol. 46, no. 10, p. R6124, 1992.

[18] T. Bakıcı, E. Almirall, and J. Wareham, "A smart city initiative: the case of Barcelona," *Journal of the Knowledge Economy*, vol. 4, no. 2, pp. 135–148, 2013.

[19] L. Sánchez, V. Gutiérrez, J. A. Galache, P. Sotres, J. R. Santana, J. Casanueva, and L. Muñoz, "SmartSantander: Experimentation and service provision in the smart city," in *Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on*. IEEE, 2013, pp. 1–6.

[20] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb. 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6740844/

[21] J. A. Galache, T. Yonezawa, L. Gurgen, D. Pavia, M. Grella, and H. Maeomichi, "ClouT: Leveraging Cloud Computing Techniques for Improving Management of Massive IoT Data." IEEE, Nov. 2014, pp. 324–327. [Online]. Available: http://ieeexplore.ieee.org/document/6978633/

[22] Z. Khan, A. Anjum, K. Soomro, and M. A. Tahir, "Towards cloud based big data analytics for smart future cities," *Journal of Cloud Computing*, vol. 4, no. 1, Dec. 2015. [Online]. Available: http://www.journalofcloudcomputing.com/content/4/1/2

[23] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, and D. Pfisterer, "SmartSantander: IoT experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, Mar. 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128613004337