Supporting Multiple Smart-City Applications based on MUSANet, a Common IoMT Middleware

Alexandre Meslin[1,2], **Noemi Rodriguez**[1], **Markus Endler**[1]

[1]Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente, 225, Gávea
Caixa Postal 38097 – 22451-900 – Rio de Janeiro – RJ – Brasil

[2]Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro (UFRJ)
Avenida Athos da Silveira Ramos, 274 – Cidade Universitária – Ilha do Fundão
21941-916 – Rio de Janeiro – RJ – Brazil

{meslin,noemi,endler}@inf.puc-rio.br

Abstract. *The MUSANet system is a three-tier middleware for smart cities implemented using InterSCity, ContextNet, and Mobile-Hub. In order to decentralize processing from the cloud, the system includes stationary layer processing in the fog and collection of mobile data in the edge. In this article, we explore the flexibility and decoupling offered by MUSANet. We present two different applications for smart cities and discuss how they can be implemented in MUSANet, showing that, using the basic infrastructure, we can build new applications without interfering in existing ones due to the low coupling between the entities that make up the tiers of MUSANet. A third application illustrates how the distribution of data processing among MUSANet layers can help reduce the network load, preserving energy.*

1. Introduction

According to the United Nations [United Nations 2918], in 2018, 55% of the world's population lives in urban areas, and this percentage is expected to increase significantly in the coming years. To manage these ever-growing cities, administrators need accurate, up-to-date city information. This information is also beneficial for the general population - knowing where traffic is jammed, when the next bus will pass, or even the air quality and temperature of some regions can make life easier in large urban centers. To provide information to administrators and the inhabitants, city and third-party organizations are continually developing applications that collect sensor data distributed throughout the city, process, and make the information available to administrators and the public. Using a single platform capable of hosting different types of applications developed by separate entities enables to optimize computing resources, increasing application efficiency, scalability, and elasticity.

Our previous work [Meslin et al. 2018] describes the MUSANet project, where we present the system architecture in detail, how it provides support for IoMT (Internet of Mobile Things), and a public transportation case study, focusing on scalability and performance studies. However, we did not discuss specific cases and how they would

be implemented in different tiers. Our goal in this paper is to discuss how our three-tier architecture allows the development of smart city applications sharing the same infrastructure, without interfering with other applications. We show that due to its modularity and low degree of coupling, the platform supports different services for mobile and stationary users, possibly designed and deployed by independent groups over the same infrastructure. To illustrate our point, we describe three different systems that can be used to implement several independent applications related to smart cities sharing the same infrastructure: a bus monitoring system, a system for user notifications, and an application to detect and monitor heat islands in a city. This approach allows the city hall offices to supply the system with data that will be administered by different organizations that would be responsible for processing and distributing the data to their users. This modularization also allows the implemented applications to share actors with well-defined tasks, without overlapping assignments, but who together perform all the functions of the systems, differently from what happens in more monolithic systems.

The remainder of this paper is structured as follows. In Section 2, we give the necessary background and definition of the MUSANet system architecture. In Section 3, we present some prototypes developed to validate the MUSANet architecture, and in all of them, we present a case study using some scenarios as examples. In Section 4, we present and discuss the results. In Section 5, we analyze some related work. In Section 6, we present some concluding remarks.

2. Overview

In this section, we present the architecture of the MUSANet middleware, shown in Figure 1, as well as the elements that make up its three tiers implemented in the cloud, fog, and mobile edge levels reflecting different degrees of abstraction, as we discuss next.

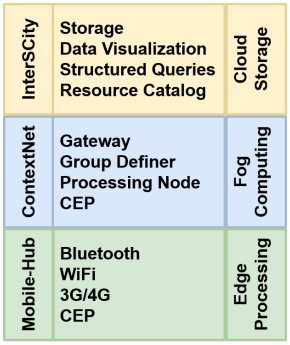


Figure 1. MUSANet three-tier architecture and its distribution.

The top tier, implemented by the InterSCity (ISC) [Batista et al. 2016], forms the cloud storage system, including a broker publisher/subscriber module. This tier is responsible for the permanent storage of structured data and high-level queries. These queries can be done either by the tier below or by external applications to MUSANet using its HTTP API. ISC provides native support for mobile devices, so all the resources of the city can be stored with references to their geographic position (coordinates) to later be retrieved with a query, for example, we can ask for a specific type of resource near any point of the city. ISC also has the role of a broker for actuators to register to receive commands. Each device, a sensor or an actuator, is registered in ISC as a city resource. Each

resource has a list of capabilities, representing what can be obtained from sensors or what the actuators can perform. No data processing occurs in this tier[1].

We implemented the middle tier using *ContextNet* (CN) [Endler et al. 2011]. To decentralize the system, we divided the ContextNet deployment in *slices* distributed on the fog. Each slice consists of a set of gateways, processing nodes, group definer, and point of access manager as follows.

Edge elements connect to the stationary infrastructure using ContextNet Gateways (CN-Gateways), which can be placed throughout the city. The *Point-of-Access Manager* (PoA-Manager) module is responsible for choosing the CN-Gateway. Together with the ContextNet protocol, which provides transparent re-connections for both the programmer and the end user, they can keep the mobile node connected to the best Gateway as it moves around the city.

A ContextNet *Processing Node* (PN) processes data received from the edge or retrieved from the cloud. PNs are programmed in Java and can count on a CEP[2] (Complex Event Processing) engine. CEP enables the PN to handle information flows using the EPL language [Luckham 2008], which is very similar to SQL. After processing or consolidating the data, the PN can either send them to the top tier to be stored in ISC or send commands to actuators at the edges.

ContextNet also allows mobile nodes to be grouped according to criteria established by system operators, such as geographic areas, as explored in this article, or by interest groups, types of employees, or even affinities. The ContextNet *Group Definer* (GD) defines the groups and associates them with mobile edge devices. This module allows the programmer to create classes containing methods that define the criteria for grouping mobile or stationary devices. Using these criteria, the GD determines the groups in which each of the edge devices is contained and informs the Gateway so that it can send group messages to each of the devices on the Internet.

To make data acquisition and processing faster, MUSANet includes a third processing layer at the edge tier of the system, where Android-based mobile devices (smartphones or tablets) running *Mobile-Hub* [Gomes et al. 2017] capture data, trigger actuators, and send and receive messages to the users. Mobile applications use Mobile-Hub to pre-process data, sending consolidated data enriched with location and timestamp context to the stationary infrastructure. Data consolidation allows power savings, which is critical for mobile devices powered by batteries since the power consumption during data transmission and reception far outweighs the consumption of the rest of the system [Estrin 2002] apud [Boukerche et al. 2003].

3. MUSANet Applications

In this section, we present three MUSANet example applications. We first introduce *RegionAlert*, a system to send context-aware messages to users. Next, we present *Bus Monitor*, a public-transportation suite compounded by *Where is my Bus?*, *Bus Arriving*, and *Delayed Bus*, to compare their programming models and their actors' roles, and then we describe the Heat Island Detector, an application developed to analyze CEP actuation

[1]There is a CEP engine currently under development for ISC.

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

on different layers.

3.1. RegionAlert

The RegionAlert application sends announcements from external entities such as the Civil Defense, Highway Patrol, or Tourism Secretary to end users. RegionAlert consists of a set of two Web Services for publishing announcements, a PN capable of retrieving published announcements, and a mobile application installed on the end-user's mobile phone implemented over the infrastructure provided by the MUSANet system.

When using RegionAlert, users can register several areas of interest created by the city administration, such as areas near their residence, their work, their most common route, the place where their elderly parents live, and more. The user's smartphone keeps the system up-to-date as to their location using the Android mobile application. Whenever a new alert is generated for the user's areas of interest or the area where the user is, the application notifies the user via text, beep, or vibration of their smartphone, according to the settings made in the mobile application by them. Alerts generated for regions outside the user's areas of interest and outside the area where the user is currently located will not be sent. A sequential-numbering message system ensures that the user will not receive repeated messages. All of this guarantees that all messages received by users are of effective interest to them. To the best of our knowledge, most alert systems implemented in large cities send messages without considering the positioning of users.

3.1.1. RegionAlert system architecture

The RegionAlert system is composed of two independent parts as shown in Figure 2: alert generation, maintained — possibly by a Civil Defense Department — as part of the city's infrastructure, and user management, maintained by a third-party organization. There is also a third part of the system: the user smartphone with the third-party mobile application running over M-Hub.

The lower tier, composed of the M-Hub, periodically sends contextualized information from the user to the infrastructure, keeping the PN updated continuously. The PN, located in the central layer, can send broadcast or unicast messages over the Internet to notify users.

To allow the city hall administration, or any other designated entity, to issue announcements, we have created the *PutAlert* Web Service, accessible through the HTTP POST method that requires the announcement start timestamp, its lifetime or end timestamp, the announcement text, and the list of areas affected by the announcement. After storing the new announcement in ISC, the PutAlert publishes in the *InterestedInNotices* topic managed by the ISC broker to let all outsourced organizations interested in announcements know that a new alert has been registered. The creation of new announcements has restricted access because it modifies the ISC database. A second web service, called *GetRegion*, is used to get the list of created regions.

The upper tier, composed of ISC, is responsible for storing alerts with their region location, expiration date, and text. Since the ISC can accept complex queries, including geographic locations, alerts can be quickly recovered.

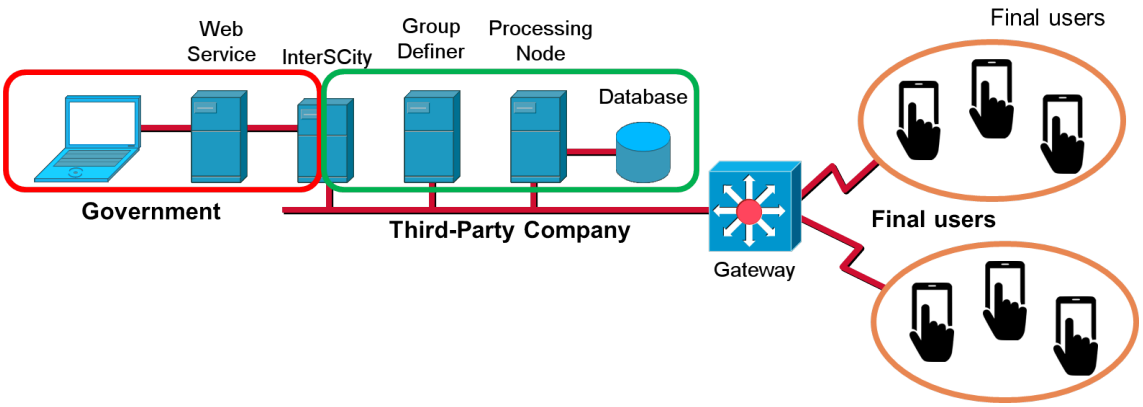


Figure 2. RegionAlert system architecture: each module (inside a delimiter) can be deployed by an independent developer.

The implementation of user management is entirely up to the outsourced organization. In our prototype, we used a MySQL database to store user data, including the areas which they would like to be warned. To be notified whenever a new alert is generated, the outsourced organization PN must subscribe to the ISC broker as an actuator with "InterestedInNotices" capability as a topic and provide a service through a public URL and a TCP port. In this case, the ISC will notify all registered resources with "InterestedInNotices" capability.

3.1.2. RegionAlert system operation

There are the following three different types of actors involved in this system: (1) the authority that registers announcements, (2) the organization that manages the users' registry and sends announcements according to their preference or location and (3) the user that will receive announcements according to their location or their preferences. Although we are describing the first two actors as separate entities, nothing prevents them from being the same organization, for example, a city hall that registers warnings and manages its users, or a university that registers warnings about its campuses and manages its students, teachers, and employees.

For each new announcement[3], the city hall administration creates a new resource in ISC with the announcement information for future reference, and publishes it as "InterestedInNotices" to the broker so that all subscriber applications become aware of the creation of a new announcement. All PNs that subscribe to "InterestedInNotices" are notified that a new announcement is available. The PNs search for the announcement in ISC using the areas of the announcement and the current timestamp as keys. The current timestamp must be used in the search so that the PN obtains only valid announcements. With the list of announcements, the PN sends the text to all the users present in the areas related to the announcement through the CN-Gateway. The PNs also search the MySQL database for users who have registered these areas as areas of interest and sends individual messages to them. Note that the PN does not need to know which users are in each group because the CN-Gateway is responsible for sending the messages to each of the

[3]the announcement source and how it is generated are not in the scope of this work.

users individually. The queries from PNs to ISC are made using HTTP.

The PN also monitors the conversation between GD and Gateway to find out when a user enters a new area. Whenever a user enters new areas, the PN queries ISC to find announcements for those areas. If any, the announcements are retrieved by the PN and sent to the user.

The end user must install the RegionAlert mobile application and the Mobile-Hub middleware. Through this mobile application, the user registers their regions of interest, informs the infrastructure of their geographical coordinates, and receives messages containing the announcements of their areas of interest or their current location. In the next version, the user will also be able to send/receive messages to/from other users in the same area.

We next present three different scenarios to exemplify the use of the RegionAlert System.

Scenario 1: In the first scenario, no alerts were created, but there are already users registered in the system moving around the city.

Suppose one or more users enter region "R". At this point, the Mobile-Hub running on the user's smartphone sends the geographic coordinates of the user to the GD. Based on the geographical coordinates, GD uses the city map divided into areas to find out which groups the user is in. Once the groups are determined, GD passes this information to the CN-Gateway and the PN. The PN is responsible for checking in ISC whether there is any active announcement (expiration time has not yet been reached) related to those areas (groups). According to this scenario, the response will be an empty list of announcements since there are no registered alerts yet.

When necessary, the Civil Defense generates an announcement message containing the announcement text, the start timestamp, the end timestamp, the list of areas (groups) involved, and a sequential number (automatically generated by the PutAlert Web Service).

The PutAlert stores this message at the ISC and publishes via the ISC broker a message to all PNs that subscribe to "InterestedInNotices" topic. The outsourced-organization's PN receives this publication and seeks for the announcement in the ISC, receiving an announcement with a list of areas. Using this list, the PNs seeks for users at the outsourced database that have marked those areas as interested and sends the announcement to all groups and all users.

Scenario 2: This scenario describes a continuation of Scenario 1, where the municipality creates a notification.

A user enters the area "R" that already has an alert message registered according to Scenario 1. The GD realizes that the user is now in a new area and sends a notification to the CN-Gateway and the PN. This notification contains a list of the areas where the user is. The PN, based on this list, queries the ISC and receives a list of valid messages for the regions where the user is. The PN sends the message to the user via the Gateway. In this scenario, it is not necessary to send the message to the groups because the other users in the groups have already received it, as described in Scenario 1. If any other user enters the region "R", the process will be repeated for this new user.

Scenario 3: This scenario considers again the user that we presented in Scenario 1 and Scenario 2, within an area that has an alert created and stored in ISC. The user has already received the announcement, as discussed in Scenario 2.

Suppose that, after receiving the announcement, the user leaves the region but returns immediately, before the lifetime of the message is exhausted. When the user enters the zone a second time, the Mobile-Hub informs the user's position and the sequential number of each received message whose lifetime has not been exhausted. The GD does not take into account that the user has already been in the region and notifies the Gateway and the PN because the GD does not store information about the user to maintain its decoupling from the third-party applications. As in Scenario 2, the PN queries ISC and receives a list of valid announcements for the user regions, but it deletes from this list all announcements the user has already received based on the sequential number. If, after deleting the repeated messages, any messages are still left, they will be sent by the PN to the user as described in Scenario 2.

3.2. Bus Monitor

The Bus Monitor system provides different applications related to public buses. In our experiments with Bus Monitor, we use the bus GPS information provided by the municipality of the City of Rio de Janeiro[4]. The information is updated every 30 seconds and contains positions, serial numbers, geographical coordinates, line numbers, and speed of all public buses in the city in JSON format. Although the time intervals between updates are considerable, the use of real bus positions allowed us to perform experiments without having to resort to vehicle movement simulators.

The Bus Monitor system offers the following three applications:

Where is my bus?: This application allows the user to determine, in (almost) real time, the location of the buses. The mobile application allows the user to filter buses by their line number. This prototype was developed to test the MUSANet infrastructure and its programming model, not adding data processing. In terms of programming, the data is captured at the bottom tier, located at the edge of the infrastructure, and sent to the fog, where the bus position data is obtained from the total data set and prepared to be sent to the storage tier located in the cloud. This application used all MUSANet's tiers in a simple way, allowing us to evaluate the performance of the MUSANet system, as presented in our previous paper [Meslin et al. 2020].

Bus arriving notification: This application notifies the user when a given bus is almost arriving at the bus stop. To be notified, users must position themselves within the area of the bus stop and use the mobile application over the Mobile-Hub. This prototype uses the Mobile-Hub located on the Android device at the edge of MUSANet to get the position of the bus and send it to the stationary infrastructure. The PN located in the fog receives the coordinates of the bus and streams it to a CEP (Complex Event Processor [Luckham 2008]) rule set to trigger a message when the bus is inside the helper area. This prototype was used to measure the delay caused by the wireless or 3G/4G network and the Internet that a client would have if it connected to a CN-Gateway far from the MUSANet slice where PN that will process the data is.

[4]Data source: `http://dadosabertos.rio.rj.gov.br/apiTransporte/apresentacao/rest/index.cfm/obterTodasPosicoes`

Delayed bus: For the operation of the Delayed Bus sub-application, it was necessary to previously catalog the amount of time that the buses use to reach the bus stops starting from the previous stop. For each pair of bus stops, the average time was obtained and a tolerance value was added. Whenever drivers arrive at the bus stops, they are notified if they are late. This notification is also stored in ISC so that the city public transport affairs office and bus company manager can analyze historical data and decide if any proactive measures are needed to improve public transportation.

3.2.1. Bus Monitor system architecture

Figure 3 shows a basic diagram of the Bus Monitor system. Mobile nodes, composed of users' smartphones, and public-transportation means as buses, trains, taxis, and more, can connect to the stationary infrastructure via Wi-Fi, 3/4/5G, or any other Access Layer TCP media.

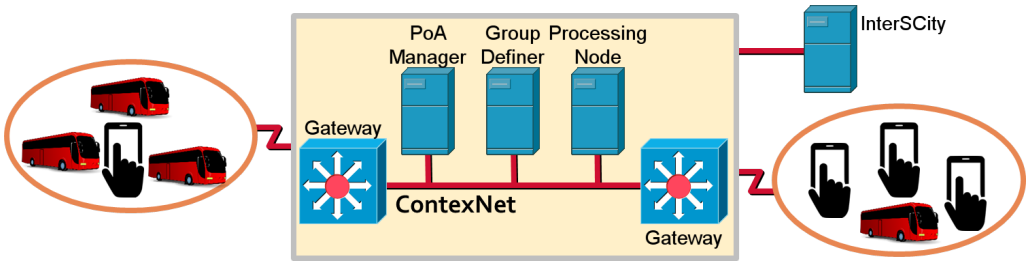


Figure 3. Bus Monitor prototype block diagram.

The implementation of the Bus Monitor system involves four different actors with well-defined roles, namely: (1) the buses that generate data through an Android mobile application installed by the municipality, (2) the public-transportation affairs office that manages the GD, creates the areas in the city and installs or supervises the installation of the Android mobile applications on the buses (3) the bus company or a third-party organization to collect the data and (4) the user who receives notifications about the buses.

Each bus reports its current position every 30 seconds using the Mobile-Hub installed on an Android embedded device — previous MUSANet simulations have shown that the system can support bus announcements every 5 seconds, considering the current fleet in circulation in the city of Rio de Janeiro. This same embedded device is ready to receive announcements or commands coming from the stationary infrastructure. The new announcements are passed on to the bus driver, while the mobile application discards repeated announcements. When moving around the city, the Mobile-Hub receives information from PoA-Manager to select the best CN-Gateway available at the moment. If the bus moves away from the current gateway or loses connection to it, it is up to the Mobile-Hub to look for another Gateway to keep the bus always connected with ContextNet on the stationary infrastructure.

A PN in a slice analyses the data received by ContextNet through each Gateway and, upon request, informs users of the current position of the bus from a specific line. This PN also feeds the CEP engine to be able to inform users that the bus is arriving at their bus stop. In addition to the area around the bus stop, a helper area was created on

the bus path before the bus stop. When the bus passes that helper area, the Mobile-Hub that is running on the Android system aboard the bus notifies the stationary infrastructure that sends a message to all users at the next bus stop.

In order to control bus schedules at bus stops, the PNs use the CEP engine to trigger a module that stores the event that will record that a delayed bus arrived at a bus stop and store the data in ISC. This event will feed another CEP stream to check if the bus delay at this point is recurring or was a one-off event. If it is recurring, this information will also be stored in ISC, and, eventually, it may also generate a message for the bus company or to the public transportation affairs office.

Whenever the bus arrives late at a bus stop, the PN sends a message to the Mobile-Hub that is running on the bus to notify the driver. This message is used to stream data to a CEP rule into the Mobile-Hub itself that will notify the PN if that same bus arrives late at several bus stops.

In our implementation, the GD and Gateway implementation for the Bus Monitor application is the same as that used in the RegionAlert notification system described above.

Management of the GD in ContextNet and the control of the virtual machines for the implementation of the PNs can be in charge of the municipality or of an outsourced organization contracted for this exclusive purpose. Once the virtual machine is created for the PN, its maintenance and programming is the responsibility of the second actor, that is, the bus company or the third-party organization.

3.2.2. Bus Monitor system operation

The bus with the Android embedded device running Mobile-Hub connects to the "nearest" Gateway of the stationary infrastructure (proximity is related to communication time rather than physical distance). As the bus travels around the city, the PoA-Manager analyzes its position and eventually reconnects the bus to a new Gateway. From that point on, the Mobile-Hub maintains ContextNet updated with its position (geographical coordinate). Whenever the GD detects that the bus has entered or exited an area, it notifies the CN-Gateway. The PN stores the last position informed of each bus to be able to provide information to the user of the "Where is my bus?" application and monitors the conversation between GD and Gateway using the Complex Event Processing engine to trigger messages to users at the next stop whenever a new event reports that a bus is present in a helper area prior to that bus stop.

Currently, the buses of the City of Rio de Janeiro are not yet using the M-Hub to connect to ContextNet, so, in order to use the bus data, we implemented a program that accesses the municipality website to obtain the bus list and their positions. This program creates a thread for each bus to send the data emulating the existence of a device running M-Hub embedded on each bus. Although we have used an emulator in part of this prototype, the data generated is real, and we include delays in the networks from the M-Hub emulator to the CN-Gateways to make the emulated part as close as possible to a real system.

3.3. Heat Islands

Heat islands [Taha 1997] are regions of a city where the ambient temperature rises a lot compared to the surrounding average temperatures. They are caused by overbuilding, poor ventilation, sparse vegetation, air pollution, among others. Through temperature sensors embedded in buses, the municipality can obtain temperature data from all over the city while monitoring the bus fleet.

3.3.1. Heat Island system architecture

In this application, the Mobile-Hub obtains external temperatures through a Bluetooth thermometer and uses them as input data to the Mobile-Hub CEP engine to search for Heat Islands in the City. If necessary, sensors can also be placed where buses do not pass, forming an ad-hoc Bluetooth network. The data captured by the sensor network are transmitted, node by node, to the bridge node, which is located close to the bus route. When a Bus with Mobile-Hub is within the range of its Bluetooth radio, the bridge node transmits the temperature information obtained by the other nodes to the bus with Mobile-Hub to process the information. This application has only one actor, the government, as shown in Figure 4.

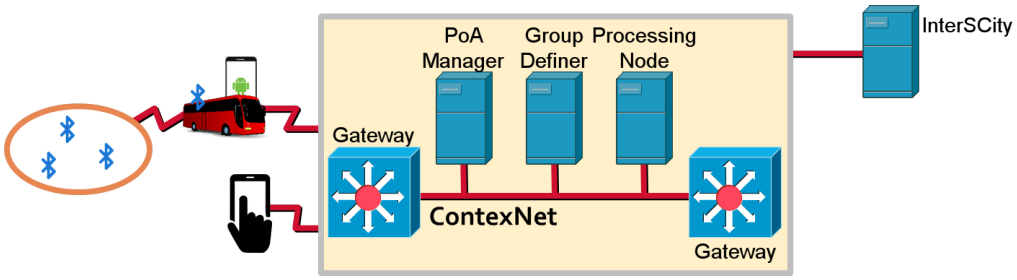


Figure 4. Heat Island prototype block diagram.

3.3.2. Heat Island system operation

Buses or other vehicles equipped with temperature sensors travel around the city. When a considerable temperature variation is detected, the CEP triggers a message for stationary infrastructure, the ContextNet. Within ContextNet, the PN receives information from the buses that detected a possible heat island border to find out its location and temperature range. If required, the PN can feed RegionAlert to send an alert message to all users located in the area stating the existence of the Heat Island, its position and temperature so that the population can avoid its effects [Tan et al. 2010] that may include mortality of the elderly as according to a study conducted by Rosenthal at Columbia University in New York [Rosenthal 2010].

Upon detection of a new Heat Island, the PN sends the data to ISC to allow for historical and immediate analysis.

If there was no local processing in the Mobile-Hub located on the buses and therefore close to the sensors, all collected data should be sent to the stationary infrastructure, implying great use of the radio, as a consequence, a high energy consumption.

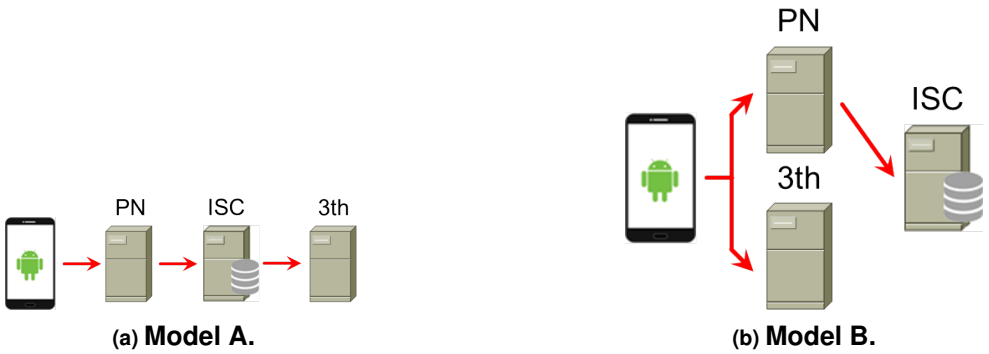
Considering a Heat Island of approximately 3000m^2 in the center of the city of Rio de Janeiro, city buses would have spent an average of 8% of their travel time inside the Island of Heat, representing approximately 24 minutes. Using the local processing based on the Mobile-Hub CEP, only temperature data when entering or leaving the Heat Island would be reported to the stationary facility, which would represent between 3 and 4 transmissions per bus during the day. If there were no local processing, all data would have to be transmitted to ContextNet, whether or not there was Heat Island, which would considerably increase the bandwidth occupation and energy consumption.

Although the bus electrical subsystem continuously power the Android device radio, and, therefore, power consumption is no longer a problem, we should consider that this is just one particular case of a MUSANet-based application. Moreover even though it is not necessary or essential to save power energy, the significant decrease in data transmission achieved with local processing allows many other applications to share the same network bandwidth. Examples of other applications are the detection of holes and other imperfections in the asphalt using impact sensor in the buses, gas levels such as O_3, CO_2, through its respective sensors, detection of burned or obstructed lamps in public lighting through light sensors, and more.

4. Results and Discussions

Through the examples cited in the section 3, we can identify two different models of application deployment by third-party organizations. The first type is most commonly found in middleware for smart cities. Its main feature is the use of the PN to collect data stored in the ISC to generate information for the end user or other applications of type M2M, as shown in Figure 5a. The third-party application can be implemented outside MUSANet, only using the infrastructure to obtain data. An example of this type of implementation is the "Where is my bus?" prototype.

A second model, exemplified by the "Bus Arriving" application, places the third-party application inside a slice of MUSANet, sharing data input with other PNs, Groups Definers, and PoA-Manager to receive information directly from the sensors before being permanently stored. Figure 5b shows this model that uses the ContextNet feature of receiving data and distributing it to PNs, GD, and third-party organization. Using this approach, a PN can receive data as soon as it is delivered to ContextNet, without going through the storage layer.



(a) **Model A.** (b) **Model B.**

Figure 5. MUSANet Application Deployment Models.

In both deployment models, the addition of more applications does not require any modification in the existing applications, facilitating the expansion of services. The choice of the model is not mutually exclusive. We can implement an application that has a PN receiving data directly from the sensors while using the information stored in ISC.

The ability to perform edge data processing, as exemplified in the "Heat Island" prototype, shows that data aggregation can significantly decrease the amount of information transmitted across the network by saving energy and optimizing bandwidth usage, two limited resources for IoT systems. Saving energy can allow battery-operated devices to have much longer battery life. Optimizing bandwidth usage, transmitting less data, allows more applications to be deployed without overloading the data network.

5. Related works

In this section, we will present and compare some related works pointing out some differences to ours.

dojot [dojot 2017] is an open-source platform for data collection of IoT-related devices, allowing the storage of large volumes of data in a distributed way in the cloud. The platform is scalable and has a usage monitoring service to increase or decrease cloud processing and storage capacity, allowing for dynamic adaptation to system requirements. Unlike MUSANet, dojot does not provide support directly for mobile devices. The acquisition of data from wireless sensor networks is not part of the system specification and is left to hardware or software from third parties, which may compromise situations where the response to events needs to be close to real time. There is no native support in dojot to handle location-related contexts, such as sending messages to groups of devices in a given geographic area.

Another example is the FIWARE platform [FIWARE Foundation e.V. 2018], an environment based on OpenStack and Docker that stores data hierarchically in a MongoDB database, supporting sensors and actuators devices. The platform also provides support to authenticate users, and user roles can be set with configurable permissions. Unlike ISC, the platform natively supports complex event analysis through the CEP for real-time event handling. Although FIWARE and dojot do not provide direct support to the locality, the implementation of a region-based warning system can be carried out, but it would be the responsibility of the programmers to implement in their application routines to verify regions, which would significantly increase complexity and coupling between systems. Sending announcements to users through the application should be done one-by-one because FIWARE and dojot do not send messages to groups separated by areas, meaning that there would be a need to implement routines to track users at all times.

Barcelona City [Bakıcı et al. 2013], 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, 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 and application layers.

The SmartSantander project (Santander, Spain) [Sánchez et al. 2013], considered

the largest testbed in the world, consists of a three-layer platform for smart cities. The upper layer is made up of servers that process and make data available to external applications. The lower layer, formed by IoT nodes installed throughout the city, is responsible for capturing sensor information and sending it to the central layer, composed of dual-stack gateways to convert protocols used in IoT, TCP / IP for transmission over the Internet. Although gateways are arranged around the city, they are used to distribute data traffic.

6. Final remarks

Comparing RegionAlert with Bus Monitor prototype, we can note that the high-level architecture is very similar since it is also composed of three independent actors with the same responsibilities in each one of the prototypes: municipality[5], third-party organization and final user. The municipality is responsible for sending data to the database in the ISC, and the third-party organization makes the information available to the end user. Unlike in RegionAlert, where the user is informed about alerts, in bus prototypes, the user must use the mobile application to request the information, except for the sub-application that informs that the bus is arriving, where the user is alerted through a message.

Using the MUSANet framework, the implementation of more services can be done without modifying the existing structure. It can be done just creating another virtual machine to host the PN with the new service system. Obviously, the creation of this virtual machine must be authorized by the municipality, which can charge for the space given.

Although the services were presented separately, but sharing the same infrastructure, we can use some of the services together to produce new applications, such as, for example, population monitoring in pandemic cases, including communication of groups according to responsibilities or locations.

Acknowledgement

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9.

References

Bakıcı, T., Almirall, E., and Wareham, J. (2013). A smart city initiative: the case of Barcelona. *Journal of the Knowledge Economy*, 4(2):135–148.

Batista, D. M., Goldman, A., Hirata, R., Kon, F., Costa, F. M., and Endler, M. (2016). InterSCity: Addressing Future Internet Research Challenges for Smart Cities. In *Network of the Future (NOF), 2016 7th International Conference on the*, pages 1–6, Buzios, Brazil. IEEE.

[5]Without loss of generality, we can consider that the role of the buses are like role of the municipality in what concerns the sending of data to feed the system since the municipality is the entity responsible for the installation (or inspection) of the bus data sending system.

Boukerche, A., Cheng, X., and Linus, J. (2003). Energy-aware Data-centric Routing in Microsensor Networks. In *Proceedings of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '03, pages 42–49, New York, NY, USA. ACM.

dojot (2017). dojot Soluções para IoT - Plataforma de Desenvolvimento para IoT.

Endler, M., Baptista, G., Silva, L., Vasconcelos, R., Malcher, M., Pantoja, V., Pinheiro, V., and Viterbo, J. (2011). ContextNet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking. In *Proceedings of the Workshop on Posters and Demos Track*, page 2, Lisbon, Portugal. ACM.

Estrin, D. (2002). Wireless Sensor Network.

FIWARE Foundation e.V. (2018). FIWARE | Open Source Platform for the Smart Digital Future.

Gomes, B. d. T. P., Muniz, L. C. M., da Silva e Silva, F. J., dos Santos, D. V., Lopes, R. F., Coutinho, L. R., Carvalho, F. O., and Endler, M. (2017). A Middleware with Comprehensive Quality of Context Support for the Internet of Things Applications. *Sensors*, 17(12):2853.

Luckham, D. (2008). *The power of events: An introduction to complex event processing in distributed enterprise systems*. Springer, Berlin.

Meslin, A., Rodriguez, N., and Endler, M. (2018). A Scalable Multilayer Middleware for Distributed Monitoring and Complex Event Processing for Smart Cities. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–8, Kansas City, MO, USA. IEEE.

Meslin, A., Rodriguez, N., and Endler, M. (2020). Scalable Mobile Sensing for Smart Cities: The MUSANet Experience. *IEEE Internet of Things Journal*, pages 1–8.

Rosenthal, J. K. (2010). *Evaluating the impact of the urban heat island on public health: Spatial and social determinants of heat-related mortality in New York City*. PhD thesis, Columbia University.

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

Taha, H. (1997). Urban climates and heat islands: albedo, evapotranspiration, and anthropogenic heat. *Energy and Buildings*, 25(2):99–103.

Tan, J., Zheng, Y., Tang, X., Guo, C., Li, L., Song, G., Zhen, X., Yuan, D., Kalkstein, A. J., Li, F., and Chen, H. (2010). The urban heat island and its impact on heat waves and human health in Shanghai. *International Journal of Biometeorology*, 54(1):75–84.

United Nations (2918). Population Facts - The speed of urbanization around the world. Technical Report 2018/1, United Nations.