SensingBus: Using Bus Lines and Fog Computing for Smart Sensing the City

Pedro Cruz*, Felipe F. da Silva*, Roberto G. Pacheco †, Rodrigo S. Couto†, Pedro B. Velloso*,
Miguel Elias M. Campista* and Luís Henrique M. K. Costa*
* Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - DEL/POLI
† Universidade do Estado do Rio de Janeiro - FEN/DETEL/PEL

***Abstract*—Different smart city applications rely on Internet of Things infrastructures to perform sensing tasks, using data to improve citizens' daily life. Nonetheless, deploying static sensing nodes over entire cities is often unpractical, since costs may become prohibitive to municipalities. If, however, sensors could move throughout the city, they would be able to cover a wider area and take advantage of opportunistic communications, reducing costs. Motivated by this idea, we propose SensingBus, a general-purpose system that collects data from sensors carried by urban buses. SensingBus is based on a three-level architecture. At the first level, sensing nodes collect and send data to the second level, consisting of fog nodes. The fog nodes pre-process and deliver data to the third level, the cloud infrastructure, which stores and makes data externally available. The fog infrastructure, on the other hand, discards defective data, compresses information, and provides secure access points between fog and cloud. To validate SensingBus, we build a prototype and perform experiments to stress the fog nodes. We verify that each one can accommodate at least 20 simultaneous sensing nodes, which is an adequate number to perform city sensing using urban bus lines of a city such as Rio de Janeiro.**

***Index Terms*—Internet of Things, fog computing, urban sensing, smart cities.**

I. Introduction

Smart cities aim at improving the quality of life of citizens by providing new public services or enhancing the efficiency of existing ones. The key idea consists of monitoring several aspects of the city, such as traffic, weather, air quality, only to cite a few, and using the collected information as the fundamental input to intelligent applications. For example, consider a typical public service: street lighting. Municipalities are interested in keeping the number of faulty light poles as low as possible, because there is a relationship between dark areas and crime events, and other problems such as car accidents. Hence, a smart city can use sensing devices to monitor light poles and detect problems in these objects. Another promising application is to predict flooding caused by storms and tell citizens to avoid areas that will potentially be affected.

The Internet of Things (IoT) paradigm can be used to collect the data of interest for each smart city application. The basic principle of IoT consists of adding communication, processing and sensing capabilities to everyday objects [1]. Nevertheless, to provide effective sensing in smart cities, one has to spread IoT devices in large geographic regions. Such strategy might prove prohibitively expensive. For example, in the case of street lighting, every light pole would be equipped with sensors and communication interfaces. Considering a big city with several thousands of light poles distributed over a large area, adding IoT devices to every light pole might impose significant deployment and maintenance costs. One alternative is to apply a Mobile Wireless Sensor Network (MWSN) that covers the entire region of interest. As sensors move through the region, the coverage area is enlarged, avoiding additional costs with IoT devices. This approach, termed as *mobile sensing* in the literature, creates a trade-off between mobility cost and the time to cover a given area [2].

One way of granting mobility to sensors is to embed them into buses, in line with the IoT paradigm. The advantage of using buses to transport sensors is threefold. First, buses cover a significant area of cities. Second, the additional cost of carrying IoT devices in buses is negligible. Third, the route of a given bus line is generally the same, with reasonably regular intervals, providing predictable coverage. Moreover, bus lines usually present some itinerary overlapping, which means that one location might be in the route of several buses. Thus, several measurements of this location can be collected, increasing the sensing reliability. In the example of street lighting, a single luminosity sensor in a bus can monitor the quality of all light poles on its route. As employing a single mobile sensor can be unreliable, different buses passing near a given light pole can measure its luminosity quality.

Typically, IoT devices have limited processing and storage capabilities, while smart city applications collect and analyze a large data amount. Thus, a solution to store and process the collected data is to use a cloud computing infrastructure [3]. IoT devices are responsible for sensing and actuation, while heavier processing tasks are performed in the cloud, which has more processing power and is able to gather all data collected in a distributed fashion. A major concern regarding this approach is the network traffic exchanged between devices and cloud, which might be high. Furthermore, security is a concern, given that IoT devices may be unable to run secure protocols to communicate with the cloud. We overcome these limitations by employing a fog computing infrastructure between IoT devices and the cloud. The role of the fog is to pre-process data before it reaches the Internet [4]. Hence, more sophisticated security protocols can be employed before the messages are sent over the Internet. Additionally, messages can

be aggregated, filtered, and compressed in the fog, reducing the traffic sent to the cloud and network costs.

This work introduces SensingBus, a system that uses the concept of mobile sensing in smart cities. To offer sensor mobility, SensingBus leverages the mobility of bus lines of public transportation systems. Thus, using the IoT paradigm, buses in SensingBus are equipped with sensors and a communication interface to collect and send the collected data to fog nodes located at bus stops. The fog sends data to the cloud, after performing preliminary processing. SensingBus is based on a three-level architecture, similarly to [4] and [5]. The first level is composed of IoT devices that collect data. The second level is the fog infrastructure which lays between IoT devices and the cloud, employed to improve security and reduce performance issues. The third level is the cloud, which receives this data and make it available to users. We also present the SensingBus prototype and a brief performance test of the fog nodes. Our results show that fog nodes can serve at least 20 simultaneous devices, and that this amount of devices is the maximum demand of 88% of the bus stops in the city of Rio de Janeiro.

II. Mobile Sensing in Smart Cities

There are different projects that employ mobile sensing to monitor smart cities. BusNet [6] is an architecture based on a delay tolerant network used to monitor road conditions. In this architecture, buses are equipped with sensors and a main bus stop is responsible for gathering and analyzing the collected data. Data arrives at the main stop by using opportunistic communications. BusNet employs sensor units, installed on buses, which send data to auxiliary nodes spread over a road network. The auxiliary nodes then send the collected data to buses traveling in the direction of the main station. Hence, buses act as sensing nodes as well as data mules. Mosaic [7] is another mobile sensing architecture, which targets measuring air quality. In Mosaic, sensor nodes are installed in vehicles and send raw data to the cloud using a GSM/GPRS module. The cloud is responsible for processing data and delivering it to end users through an API. One of the cloud tasks is to calibrate the received information by removing inaccurate data. This calibration is performed using Artificial Neural Networks (ANN) and Support Vector Machines (SVM).

SensingBus communication architecture lies between BusNet and Mosaic approaches. We employ opportunistic communications between buses and fog nodes which are located at bus stops. This approach is similar to the communication between sensor units and auxiliary nodes of BusNet. Nevertheless, fog nodes receive data and send it directly to a cloud, instead of using data mules. In Mosaic, the information is sent from sensing nodes directly to the cloud. Hence, differently from Mosaic, we employ a fog infrastructure between sensing nodes and the cloud. This fog can perform pre-processing, such as data calibration implemented in Mosaic.

The project SmartSantander [8] and the work of Alsina-Pagès *et al.* [9] also employ vehicles to perform urban mobile sensing. SmartSantander employs sensor nodes attached to urban vehicles in order to gather data about the city, together with other data sources. The project follows a three-tier architecture, with IoT, Gateway and Server tiers. The work of Alsina-Pagès *et al.* analyses a WSN with urban buses for noise monitoring. Alsina-Pagès *et al.* [9] list the requirements of this network and evaluate different options to perform signal processing using sensor nodes. The main difference between SensingBus and both projects is that the gateway nodes in SensingBus can pre-process data, composing a Fog level and enabling new services and architectural possibilities.

There are some works employing mobile sensing to collect data and perform scientific analysis. Some examples address issues related to air pollution mapping. In these projects, data is collected to build models and infer the air quality in a city. Since their main focus is data processing, they do not propose an architecture to store and make information available. Opensense [10] is an example, which uses buses in Lausanne to acquire data and propose models to estimate polluted locations. Another approach focused on data processing is presented in [11], where sensors are installed in Google Street View cars in Oakland. Their work provides different conclusions about air quality in this city, as well as generic models or technical information that can be applied in other cities. Finally, the work in [12] employs Google Street View cars to detect gas leaks.

The above-mentioned works confirm the potential of mobile sensing for smart cities and indicate that an integrated strategy for city sensing can prove useful. Hence, SensingBus can improve the mentioned approaches by applying the Fog paradigm into the mobile urban sensing with low-cost equipment. This strategy can enable new applications, reduce costs and improve quality of service.

III. SensingBus

The main goal of SensingBus is to provide a monitoring platform which facilitates the development of smart city applications. SensingBus implements the whole monitoring process, which comprises data sensing, transmission, storage, and service. The main SensingBus functionalities are:

- ***Data gathering*:** SensingBus gathers data along the path of participating buses, i.e., the ones equipped with a sensing node;
- ***Data pre-processing*:** SensingBus pre-processes data before transmitting it through the Internet. For instance, inconsistency is filtered, so the links to the Internet are not overused. Additionally, using signed certificates, SensingBus avoids that unauthorized devices can publish or modify the data before it reaches users;
- ***Data transmission*:** gathered data is transmitted from buses to users. Users must be able to ensure that data is not fake, i.e., it was gathered by SensingBus and was not modified during transmission.
- ***Data storage*:** all data is stored in a single database in the cloud;
- ***Data service*:** urban data collected by SensingBus is made available to users.

To provide such functionalities, the architecture of SensingBus is divided in three levels: Sensing, Fog, and Cloud.

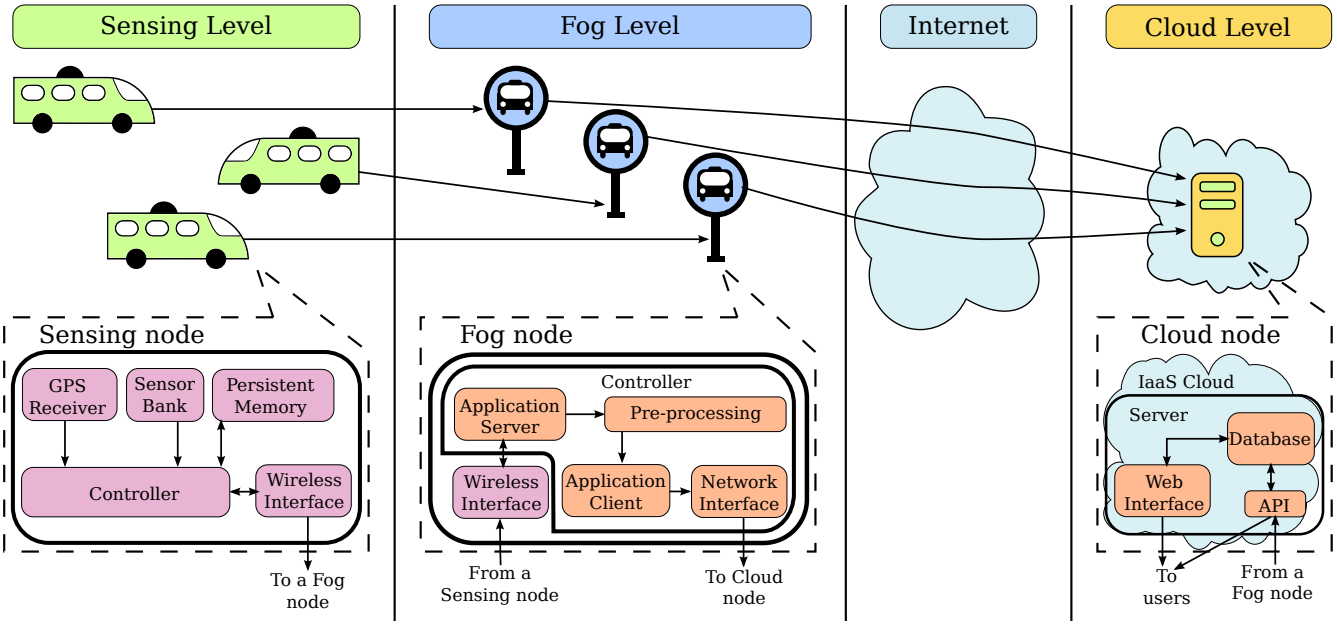


Fig. 1. The three-level architecture of SensingBus.

The Sensing level is composed of Sensing nodes located in the vehicles. This level performs data gathering. Data is then sent to the Fog level, consisting of fixed nodes installed into strategic spots around the city, such as bus stops. The Fog level pre-processes the data, as a fog element in the network. The Cloud level runs on a cloud service and receives, stores, and processes the data. The Cloud level also gives end users access to data. The architecture of SensingBus and its nodes is shown in Fig. 1.

A. Sensing level

The Sensing level is responsible for gathering data about the city and sending it to Fog nodes. Along with the sensed data, the Sensing node registers the timestamp and geographic coordinates of each sample. This level is composed of the set of all Sensing nodes located in the buses.

The Controller of the Sensing node (Fig. 1) manages all tasks of the node. At a specific sampling rate, the Controller reads the coordinates and time indicated by the GPS Receiver and also reads the measurements of every sensor in the Sensor Bank. The coordinates, time, and measurements are associated and written into the Persistent Memory. At every iteration, the Controller also queries the Wireless Interface to check whether a connection is established with a Fog node. The Sensing node connects to the Fog node using a private network, where all the other devices are trusted. Whenever a connection is established, the Controller sends all the data stored in the Persistent Memory to the Wireless Interface. This data is sent to a Fog node, and can thus be deleted from the Persistent Memory in the Sensing node. The size of the Persistent Memory must be enough to hold all data gathered between any two consecutive encounters of the Sensing node with a Fog node.

One design objective of SensingBus is to have inexpensive Sensing nodes because, ideally, every urban bus in the city will carry one. Thus, Sensing nodes are simple and constrained in terms of computational resources.

B. Fog level

The Fog level follows the fog computing paradigm. Fog nodes are located at the edge of the network. Their computational power is at an intermediate level between Sensing nodes and nodes in the Cloud level, which are cloud servers. Basically, Fog nodes receive raw data from the Sensing level, pre-process, and send it over the Internet to the Cloud level. The pre-processing is fundamental to our system since it can provide important functionalities that improve performance and security. These functionalities include data aggregation, data compression, cryptography, and others.

Fog nodes (Fig. 1) act as both application clients and servers, depending on the context. From the viewpoint of Sensing nodes, Fog nodes are application servers receiving data. Additionally, Sensing nodes also view Fog nodes as access points, since Fog nodes create private networks for communication between Fog and Sensing nodes. On the other hand, to the nodes in the Cloud level, Fog nodes are application clients. They connect to the Cloud level using the Internet and forward data received from the Sensing nodes.

Communication between Sensing nodes and Fog nodes happens on a private network, therefore, a certain security level is assured. Since messages between Fog nodes and Cloud nodes travel over the Internet, it is important that Fog nodes use protocols that implement authorization and data integrity. For this reason, it is expected that every Fog node has a certificate, signed by a Certificate Authority known and trusted by Cloud nodes.

C. Cloud level

The Cloud level is the final destination of collected data. Inside it, data is stored, processed, and made available to users.

There are many reasons for having these tasks performed by a cloud service, but the most important are the elasticity and availability of the cloud. Elasticity is important because we expect intensive algorithms to run over data from times to times. Availability is important because SensingBus might act as the core of fundamental services, such as flood warnings.

The web server of the Cloud node waits for requests from Fog nodes and users. Fog nodes can only add data to the database, whereas users can only read data from the database. The Cloud node consists of a virtual machine running on a distributed IaaS cloud. This arrangement provides elasticity, allowing the processing and storage of the Cloud node to grow and shrink, on demand. Moreover, availability is favored by the distributed aspect of the IaaS cloud. Different sites of the distributed cloud can replicate the same Cloud node, improving the resilience of the system.

Cloud nodes must hold a certificate signed by a Certificate Authority trusted by the nodes of the Fog level, in such a way that Fog nodes know if they have their messages intercepted and even discarded. Additionally, Cloud nodes must be capable of checking the certificates presented by Fog nodes. This prevents malicious devices from inserting data into the Cloud node database.

The main limitation of SensingBus is related to its spatial and temporal coverage. The spatial coverage is limited because SensingBus can only cover the streets that are visited by buses. The temporal coverage is limited because buses travel through the city, covering different areas at each moment. Therefore, SensingBus can serve as a main source of data, as other networks cover the areas SensingBus cannot reach.

IV. Prototype

We build a prototype for SensingBus in order to confirm its feasibility and scalability. The equipment and software used in the Sensing, Fog, and Cloud nodes are listed in Table I. Our source code is available at https://github.com/pedrocruz/sensing_bus. The next sections explain in detail the different parts of the prototype.

A. Sensing nodes

Sensing nodes employ an Arduino UNO as Controller, due to its low cost. The Wireless Interface is an ESP8266, a programmable IEEE 802.11b/g/n interface. Physically, this node is composed of two parts: one is kept inside the bus and the other is installed outside the vehicle. This division is performed since some modules must be exposed to the environment, such as sensors and transmitters, while other components must be protected from harsh environment conditions. For example, rain can harm some electronic equipment, but a rain sensor must be exposed to it. The part inside the vehicle contains the Controller, GPS Receiver, and Persistent Memory. The GPS Receiver and Persistent Memory are on the same board, the most expensive in this node. The external part contains the Wireless Interface, GPS Receiver antenna, and Sensor Bank. The external part is protected by an acrylic case measuring $7 \times 7 \times 4$ cm, on the rooftop of the bus. Since sensors need to detect environmental issues, the protective case has transparent surface and some holes, allowing sensors to effectively sense the environment. The effect of the protective case and mobility on the measurements were tested on a previous work [13], using a similar component layout. The following functionalities are currently implemented in the Sensing nodes:

- **Data gathering:** data is gathered by sensors in the sensor bank;
- **Data temporary storage:** data is stored until a connection with a Fog node is possible;
- **Data delivery:** data is delivered to a Fog node, when a connection is available.

Taking advantage of the programmable Wireless Interface, we implement a simple protocol for the communication between the Controller and the Wireless Interface. In this protocol, the Controller asks periodically if there is a connection and the Wireless Interface answers. When the answer is positive, the Controller sends enough data to fill the buffer of the Wireless Interface, waits for data to be sent and initializes a new iteration, asking again if there is a connection.

As an implementation issue, messages have constant values per node, such as a header, indicating the sensors installed in the bank. Since the SRAM memory of Arduino Uno is limited, we treat all the constants as strings and store them in the flash memory. In the first implementation on ESP8266, every message sent would open a TCP connection and close it after the message was sent. Even though this approach saves lines of code, every new connection creates new objects that are not discarded immediately when the connection closes, causing instability to ESP8266. To solve this issue, the connection status is kept and ESP8266 performs several requests over the same connection.

The Wireless Interface holds the SSID and password of a WPA2-protected network created by the Fog nodes. The Wireless Interface searches periodically for a wireless network with the pre-configured SSID, and answers queries from the Controller. After the Wireless Interface signals the Controller about a new connection, the Controller only sends application-specific data to the Wireless Interface. The Wireless Interface prepares the HTTP headers and executes a POST method to the Fog node. According to our experiments, this approach is two times faster than preparing the HTTP headers inside the Controller. The codes used in this test can be found in folder "sensing/tests" of the repository.

B. Fog nodes

The equipment used in Fog nodes are shown in Table I. Raspberry Pi II Model B is chosen as Controller because of its computing power, size, costs, and available interfaces. WiPi is chosen as the Wireless Interface due to its costs and because of the number of simultaneous connections that it supports. Some other options with greater communication range were considered but discarded, because of the low number of simultaneous connections supported. The effect of simultaneous connections is further discussed in the following sections.

TABLE I
EQUIPMENT AND SOFTWARE USED IN THE SENSINGBUS PROTOTYPE.

Level node	Module	Equipment	Manufacturer/Publisher	Type
Sensing node	Controller	Arduino UNO R3	Arduino	Hardware
	GPS Receiver	GS-96U7	Guangzhou Xintu	Hardware
	Persistent Memory	GS-96U7	Guangzhou Xintu	Hardware
	Wireless Interface	ESP8266	Espressif	Hardware
	Sensor Bank	Humidity Sensor DHT11	DFRobot	Hardware
		Temperature Sensor DHT11	DFRobot	Hardware
		Light Intensity Sensor GL5528	GBK Robotics	Hardware
		Rain Intensity Sensor GL5528	GBK Robotics	Hardware
Fog node	Controller	Raspberry Pi II model B	Raspberry Pi Foundation	Hardware
	Wireless Interface	WiPi	Element14	Hardware
Cloud node	IaaS Cloud	OpenStack	OpenStack Foundation	Software
	Server	Apache 2.4.18	Apache Software Foundation	Software
	Database	MySQL 5.7	Oracle	Software
	Web Interface	Django	Django Software Foundation	Software
	WAPI	Django REST Framework	Tom Christie	Software

The Wireless Interface of Fog nodes acts as an IEEE 802.11 access point protected with WPA2 (Wi-Fi Protected Access 2). The SSID and password are configured into the Controller. In the Controller, an HTTP server implemented in Python serves incoming requests and pre-process data before sending it to the Cloud node. Fog nodes have the following pre-processing functions currently implemented:

- **Error detection:** data is checked for inconsistencies and defective data is discarded;
- **Data concentration:** data is stored and sent periodically. We use a single connection at the end of each interval, reducing the number of connections to Cloud level;
- **Data compression:** data is compressed before it is sent, reducing the traffic to the Internet, and, consequently, the costs with subscriptions.

For every request, the Fog node Controller checks data and discards inconsistent measurements. After that, data is accumulated on a concentration queue. At regular intervals, data is compressed and sent to the Cloud node. To send data, the Fog node acts as HTTPS client, getting authorization by presenting its certificate to the Cloud node.

C. Cloud node

The prototype of Cloud node is implemented in software. The software used is shown in Table I. An Apache (http://httpd.apache.org/) server runs on a virtual machine, instantiated on an IaaS cloud. The Apache server executes a Django (https://www.djangoproject.com/) application, that relies on MySQL (https://www.mysql.com/) to store data, and on Django REST Framework (http://www.django-rest-framework.org/) to create API endpoints. The Cloud node exposes an URL for data insertion. This URL is protected by Apache, which only authorizes clients if they present valid certificates, generated by a trusted Certificate Authority. Therefore, on SensingBus as a whole, data integrity and authorization between Sensing level and Fog level rely upon WPA2, whereas between Fog and Cloud levels, data integrity and authorization are assured by HTTPS. The Cloud node prototype also provides API endpoints for querying data. Data can be queried by date, time, location and sensor type. Thus, users can use SensingBus to develop their own applications. The documentation of the API can be found together with the available code.The following functions are currently implemented in Cloud nodes:

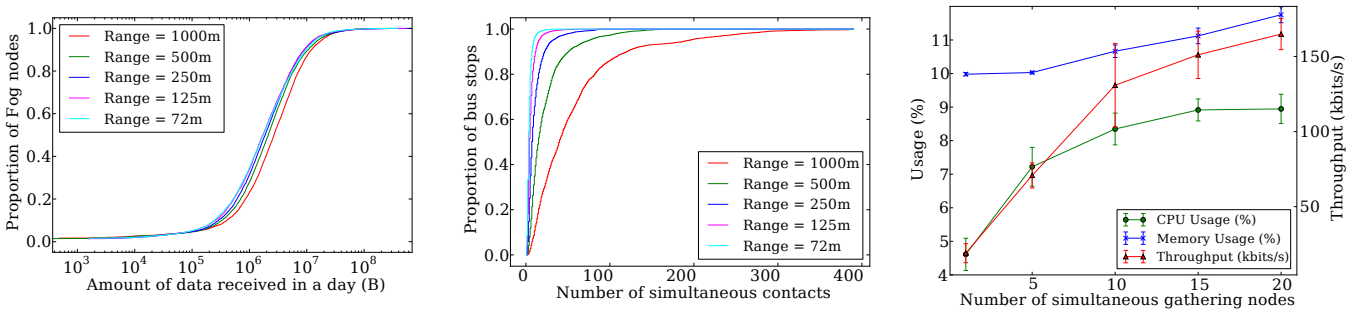
- **Data insertion endpoints:** API endpoints are exposed for data insertion;
- **Data query endpoints:** API endpoints are exposed for data query. Data can be queried by date, time, location and sensor type.

The communication between different nodes is performed using HTTP/HTTPS and users fetch data using RESTful APIs. This ensures uniform interfaces, to deal with the heterogeneity of devices and applications.

D. Performance Analysis

We conduct a number of experiments to test the scalability of our Fog node prototype. We aim to evaluate the number of requests a single Fog node has to serve simultaneously. To estimate this number, an analysis of two real datasets is performed. The first dataset contains the positions of all bus stops in Rio de Janeiro (http://data.rio/dataset/pontos-de-parada-de-onibus). The second dataset contains the positions of all buses in Rio de Janeiro, refreshed every minute (http://data.rio/dataset/gps-de-onibus). We collected the positions of all buses throughout 8:00 AM and 10:00 PM of November 30, 2016. The early and late hours of the day are discarded because a great number of buses is parked, creating distortions in our evaluation.

We assume that every bus carries a Sensing node and every bus stop has a Fog node installed. For every minute, the positions of buses and bus stops are compared. If the distance between a bus and a bus stop is shorter than a given threshold that represents the communication range, we consider that the Sensing node in the bus can send data to the Fog node on the bus stop. We define five different communication ranges: 1000, 500, 250, 125, and 72 meters. Using this dataset, it is possible to evaluate the amount of data received by every Fog node. Fig 2(a) illustrates the cumulative distribution of the



(a) Cumulative distribution of data received by each Fog node in a day, for different communication ranges.

(b) Cumulative distribution of maximum number of Sensing nodes served by each Fog node, for different communication ranges.

(c) Stress tests of a Fog node prototype.

Fig. 2. Analysis of the Fog nodes of the prototype.

amount of data received by each fog node throughout a day. It is possible to note that, for all ranges, less than 40% of Fog nodes receive less than 1 MB of data. Also, for every range, 10% of the Fog nodes concentrate more than 47% of the traffic. In order to test the scalability of the Fog node prototype, we evaluate the maximum number of Sensing nodes served simultaneously by each Fog node, during the day. Fig. 2(b) shows the cumulative distribution of the maximum number of Sensing nodes served simultaneously by each Fog node, for different communication ranges. According to Rubinstein *et al.* [14], the range of IEEE 802.11g in traffic speed is inferior to 250 m. Since our prototype can use IEEE 802.11b/g/n, we adopt 250 m as a reference. In Fig. 2(b), it is possible to note that, for a communication range of 250 m, around 88% of Fog nodes serve less than 20 Sensing nodes simultaneously. Therefore, we adopt 20 as the maximum number of Sensing nodes to use in the stress test.

We use the same datasets to verify the intercontact times of buses with bus stops, considering a communication range of 250 m. We observe that, in more than 98% of the samples, a bus waits less than 10 minutes between contacts. Therefore, we configure every Sensing node in our stress test with data equivalent to 10 minutes of sensing. Given that the sensor bank in our prototype generates 53 bytes every second, 10 minutes of sensing produces 31,800 bytes. In our implementation, this is equivalent to 15 HTTP POSTs from the Sensing node to the Fog node. The data generation rate of our prototype indicates that the Persistent Memory has to store less than 5 MB in order to store data for a whole day. In addition, the maximum expected time a bus travels with no Fog contact is much shorter than a day.

To simulate the simultaneous arrival of Sensing nodes in the communication range of a Fog node, we simultaneously turn on one, five, ten, fifteen, and twenty Sensing nodes within range of the Fog node. After Sensing nodes are on, we wait all data to be received by the Fog node and transmitted to the Cloud node of the prototype. For each number of Sensing nodes, the test is repeated 30 times. Throughout the tests we did not observe any instability in our nodes. The codes used in this experiment are available in the folder "fog/stress_tests" of the repository. Fig. 2(c) shows the results for the average memory and CPU usage since the first POST arrives at the Fog node, until the last POST arrives at the Fog node, for all the participating Sensing nodes. Measurements are presented with 95% confidence interval. On Fig. 2(c), one can observe that the Fog node can serve up to 20 Sensing nodes without exhausting its memory and CPU resources. It is worth noting that the average throughput obtained by each Sensing node drops as the number of Sensing nodes increases, as a consequence of competition for the medium. In a previous work, we have studied the intercontact times between buses and bus stops [15]. Considering the data generation rate of the sensor bank, we conclude in [15], that the throughput per sensing node must be at least 3 kbps to deliver all data gathered in a single contact opportunity. Our experiments show that our prototype suits such requirements, even in the worst case, with 20 simultaneous sensing nodes sharing the same fog node.

V. Conclusion

This paper presented SensingBus, a system for smart city sensing based on urban buses. SensingBus combines the concepts of IoT, fog, and cloud computing in order to collect, preprocess, process, store, and serve sensed data. One of the main goals of SensingBus is to provide a low-cost general-purpose solution that collects and provides to users information of different locations of a city.

We have also demonstrated the feasibility of SensingBus through the results obtained using a prototype, built using low-cost hardware. Those results analyze the scalability of the SensingBus prototype. Using data from real bus lines, we have shown that the Fog node prototype supports at least 88% of bus stops in Rio de Janeiro. Note that in a real implementation we may not install fog nodes in all bus stops. Hence, being able to choose 88% of the bus stops is more than enough.

As future work, we plan to build a pilot version of the system over the internal bus fleet of UFRJ campus. We are also investigating promising end-user applications that can benefit from data gathered from SensingBus, as well as a list of sensors that should be added to the Sensor Bank. Another plan is to use fog nodes to coordinate a cooperation between buses, to eliminate, or at least reduce, the amount of duplicate data.

ACKNOWLEDGMENTS

This work was partly funded by FAPERJ, CAPES, CNPq, and grants #15/24494-8 and #15/24490-2, São Paulo Research Foundation (FAPESP)

REFERENCES

[1] J. Gubbi *et al.*, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[2] B. Liu *et al.*, "Mobility improves coverage of sensor networks," in *6th ACM MobiHoc'05*, May 2005, pp. 300–308.

[3] K. Xu, Y. Qu, and K. Yang, "A tutorial on the Internet of Things: From a heterogeneous network integration perspective," *IEEE Network*, vol. 30, no. 2, pp. 102–108, 2016.

[4] F. Bonomi *et al.*, "Fog computing and its role in the Internet of Things," in *ACM Workshop on Mobile Cloud Computing (MCC)*, Aug. 2012, pp. 13–16.

[5] W. Li *et al.*, "System modelling and performance evaluation of a three-tier cloud of things," *Future Generation Computer Systems*, vol. 70, pp. 104–125, May 2017.

[6] K. D. Zoysa *et al.*, "A public transport system based sensor network for road surface condition monitoring," in *Workshop on Networked Systems for Developing Regions (NSDR)*, Aug. 2007.

[7] W. Dong *et al.*, "Mosaic: Towards city scale sensing with mobile sensor networks," in *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, Dec. 2015, pp. 29–36.

[8] L. Sanchez *et al.*, "Smartsantander: Iot experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[9] R. M. Alsina-Pagès *et al.*, "Design of a mobile low-cost sensor network using urban buses for real-time ubiquitous noise monitoring," *Sensors*, vol. 17, no. 1, p. 57, 2016.

[10] A. Marjovi, A. Arfire, and A. Martinoli, "High resolution air pollution maps in urban environments using mobile sensor networks," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Jul. 2015, pp. 11 – 20.

[11] J. S. Apte *et al.*, "High-resolution air pollution mapping with google street view cars: Exploiting big data," *Environmental Science & Technology*, vol. 51, no. 12, pp. 6999–7008, Jun. 2017.

[12] J. C. von Fischer *et al.*, "Rapid, vehicle-based identification of location and magnitude of urban natural gas pipeline leaks," *Environmental Science & Technology*, vol. 51, no. 7, pp. 4091–4099, Mar. 2017.

[13] P. Cruz *et al.*, "On the accuracy of data sensing in the presence of mobility," in *7th NoF'16*, Nov. 2016.

[14] M. G. Rubinstein *et al.*, "Measuring the capacity of in-car to in-car vehicular networks," *IEEE Communications Magazine*, vol. 47, no. 11, Nov. 2009.

[15] P. Cruz, R. S. Couto, and L. H. M. K. Costa, "An algorithm for sink positioning in bus-assisted smart city sensing," *Future Generation Computer Systems*, 2017.