UrbanSense: an Urban-scale Sensing Platform for the Internet of Things

Yunior Luis*, Pedro M. Santos*, Tiago Lourenço, Carlos Pérez-Penichet[†], Tânia Calçada, Ana Aguiar*
Departamento de Engenharia Electrotécnica de Computadores, Faculdade de Engenharia, Universidade do Porto
*Instituto de Telecomunicações
[†]Uppsala University
Email: {jrluis, pedro.salgueiro, up20140260}@fe.up.pt, cpp@it.uu.se, {tcalcada, anaa}@fe.up.pt

Abstract—**A critical step towards smarter and safer cities is to endow them with the abilities to massively gather a wide variety of data sets and to automatically feed those data to decision support tools and applications that leverage artificial intelligence. We present UrbanSense, a platform deployed on the streets of a mid-size European city (Porto, Portugal) to collect key environmental data. The main innovations of UrbanSense are (1) design for affordability and extensibility, (2) its ability to leverage heterogeneous networks to send the data to the cloud (using both real-time and delay-tolerant communications), and (3) its Internet of Things integration to expose the data streams to smart city tools and applications. Beyond discussing the design choices, we present operational results for 6 months of operation and give a detailed account of the challenges faced by the successful deployment of urban sensing technologies in the wild.**

I. Introduction

There is a worldwide movement towards improving sustainability and livability of urban environments by turning them smarter: using data and artificial intelligence to support decision-making processes in the urban context [1]. Applications range from enabling citizen engagement in city-wide urban planning to data-based decision support for policy makers [2]. One critical building block for such smart methods is automatically gathering fine grained urban data [3]. Many cities have had sensors in buildings and streets for decades, but most of this data is gathered during temporally limited measurement campaigns and/or collected manually with significant human effort. Additionally, data thus gathered is often dispersed throughout institutions/departments and not publicly available through a consistent interface.

In the city of Porto, three urban scale testbeds have been deployed under the EU project Future Cities[1]: a vehicular network, a crowd-sensing platform and an environmental sensor network platform. The vehicular network, BusNet, is formed by 600 nodes that support vehicle-to-vehicle and vehicle-to-infrastructure communications. The crowd-sensing tool SenseMyCity enables collection of personal mobility and questionnaire input from a growing set of volunteers. The third testbed, an environmental sensor platform called UrbanSense, is presented in this article.

UrbanSense is an urban-scale infrastructure for monitoring environmental phenomena. It consists on static Data Collecting Units deployed at relevant locations. Each unit is a hub for a series of sensors measuring weather, noise, and air quality parameters. Data communication occurs over multiple backbone possibilities: a fixed fiber ring, cellular, or the vehicular network as a delay tolerant network. Data is gathered in a cloud server which serves as an IoT gateway. We present an initial deployment of 23 sensor units for a period of 6 months and performance metrics for a set of 15 units.

The remainder of this paper is organized as follows. Related literature is presented in the following section. Section III provides an overview of the UrbanSense platform. A description of the DCU and sensor hardware is presented in section IV. Section V presents our heterogeneous communications infrastructure, and Section VI describes the software and data architecture. In Section VII we discuss some implementation issues, and Section VIII evaluates selected performance metrics of the platform. Final remarks are drawn in Section IX.

II. Related Work

The interest in smart cities has emerged in recent years fueled by advances in communication and sensing technologies. The work of [4] provides an overview of communication architectures for Internet-of-Things platforms oriented for smart cities. Academic testbeds to carry out research on large-scale ad hoc networks composed of embedded devices date from mid-2000s, of which we highlight on CitySense [5] (Cambridge, MA), RoofNet [6] (Berlin) and MoteLab [7]. Today there is a growing number of IoT deployments environmental sensing in cities all over the world. A discussion on the evolution of low-cost sensors for wide-scale environmental sensing can be found in [8]. Clairity [9] is a platform deployed in the MIT campus dedicated to sensing air quality. OpenSense [10] is a set of meteorological sensors deployed in Zurich, Switzerland, that has recently started to integrate information from smartphones. Open source hardware solutions for monitoring systems are provided by the Acronet project [11].

III. UrbanSense Overview

The UrbanSense platform is composed of three main components: the Data Collection Units (DCU) that perform city-wide sensing, a backoffice for data storage and interfacing with

[†] Faculdade de Engenharia, Universidade do Porto, at the time of his contribution.

[1] http://futurecities.up.pt/

978-1-5090-1846-8/16/$31.00 ©2016 European Union

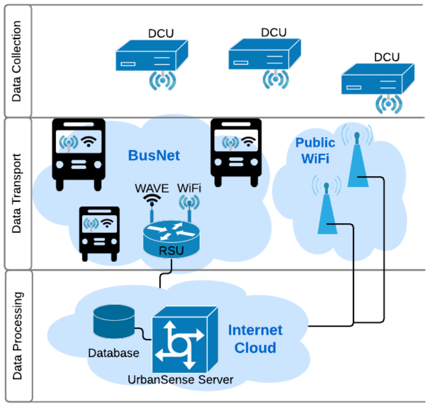


Fig. 1. Overall architecture of platform.

external clients, and a communications backbone to gather data from all DCUs to the backoffice. The global architecture of UrbanSense is shown in Figure 1.

The Data Collection Unit is the fundamental unit of the UrbanSense platform. It is a sensing hub, containing sensors for weather, environmental and noise parameters. Figure 2 presents the external aspect of a deployed DCU and a breakdown of its components. DCUs are placed throughout the city at locations identified by Environment and Acoustics experts as representative of typical urban environments.

Data gathered by DCUs is carried to the UrbanSense backoffice server via one of three available backhaul networks: metropolitan fiber ring, cellular network, or vehicular delay tolerant network (DTN). The metropolitan fiber ring is available from the municipality for free, but does not cover every DCU location. The vehicular DTN service is provided by buses and municipality vehicles equipped with WiFi hotspots; it is also free, and covers several additional DCUs. Cellular networks have wide coverage but involve high cost; in our case, cellular is only used in 2 of the 23 deployed DCUs.

The backoffice server listens for DCU connections, hosts a central database, and supports IoT gateways to access the data. Data collected from DCUs is kept in a central database for archival purposes and sharing with external entities via standard IoT publish-subscribe middleware, such as FI-WARE [12] or ETSI M2M [13]. Opting for a gateway architecture instead of having all devices visible in the IoT is a conscious design choice driven by the minimum capabilities of the available backhaul technologies, as will be explained in Section V.

IV. DCU and Sensors Hardware

The DCU has three main hardware components: the sensor board, the processing board and the control board. These components are also shown in Figure 2 (sensor board on left side, control board atop processing board on right side). We now describe their function and the available sensors.

A. Sensor board

The Sensor Board contains most of the environment monitoring sensors and support circuitry. A set of predefined sensors, detailed in Section IV-D, may be connected to this board. The sensor board handles the details of analog-to-digital conversion in the case of analog sensors, along with any signal conditioning that may be necessary. The sensor board is connected to the rest of the system by a flexible cable. This solution allows to fit the sensors inside a special enclosure through which atmospheric gases can circulate, and improves overall sensor exposure to target phenomena while protecting the remaining electronic components from the elements.

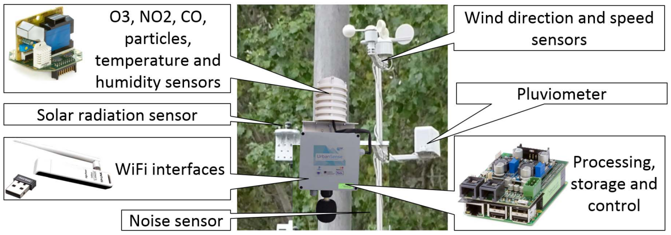

O3, NO2, CO, particles, temperature and humidity sensors
Wind direction and speed sensors
Pluviometer
Solar radiation sensor
WiFi interfaces
Noise sensor
Processing, storage and control

Fig. 2. External aspect and breakdown of a DCU.

B. Processing board

The processing board is at the core of the DCU, as it interfaces collection and storage of sensor data and the communications infrastructure. It is an off-the-shelf single-board computer (SBC); currently a Raspberry Pi 2 is used. The processing board requests measured sensor values from the control board, and manages the communication with the backoffice server via the assigned communication backhaul. The processing board also performs temporary data storage using a local database. The design option was motivated by the need to provide consistent and robust persistent storage for the collected data during relatively long periods of lack of connectivity. This applies particularly if the assigned communication backhaul is the vehicular DTN. Data must be stored while waiting for a transfer opportunity and, since the amount of waiting cannot be exactly quantified beforehand, we opted to store data persistently. In addition, this design makes data storage very robust against power outages and similar incidents.

C. Control board

The control board handles the low-level details of interacting with the actual sensor equipment. It contains a microcontroller to interface all sensors and the processing board. It also contains power electronics that provide stable power supply to all system components. The design option of creating a control board that interfaces the sensors and the processing board is meant to provide abstraction to the processing unit that is used. At any point, the current single-board computer can be switched by a new one without the need of re-implementing drivers for each sensor; just the communication protocol with the control board needs to be ported.

D. Sensors

A DCU may include a wide variety of sensors in several combinations. Table I summarizes the sensors and their most relevant characteristics, and Table II shows an estimation of the daily data volume generated by the platform for the current deployment of 23 sensor units. The sensors have a range of

TABLE I
List of sensors and their characteristics

Sensor	Digital	Bus	Heating element	Power consumption
Temperature & RH		I^2C		$5mW$
Precipitation		1 channel		$< 6mW$
Wind speed		1 channel		$< 6mW$
Wind direction		1 channel		$< 6mW$
Luminosity		I^2C		$\sim 1mW$
Sonometer		UART		$< 500mW$
Solar radiation		1 channel		self-powered
Particles		2 channels		$\sim 450mW$
CO		1 channel		$15-150mW$
NO_2		1 channel		$< 50mW$
O_3		1 channel		$< 100mW$

TABLE II
Daily data volume produced by UrbanSense

Meteoreological station	Sensors	Sampling rate	Packet size (bytes)	MB/day	Quantity
	All	1 min	668	29.00	14
	Noise	1 sec	330		
	All	1 min	818	29.21	9
Total	721.56 MB/day				

setup and communication requirements. Some sensors require specific conditioning, bias circuits, or even powering an heating element. Sensors may have digital or analog interfaces: the first set may require specific communication protocols, whereas others return analog signals that need to be quantized. To enable the deployment of a large number of DCUs covering an urban area, a design requirement was to use low-cost sensors. This raised the need to develop practical strategies to calibrate all sensors; this is discussed in Section VII.

V. Communication Architecture

A. Communication Backhauls

The UrbanSense platform relies on multiple communication backhauls to transport sensor data from DCUs to the backend server: fiber, cellular or vehicular DTN. The DCU connects to backhaul hubs via WiFi. The fiber ring is accessible via municipality-run WiFi hotspots that offer free Internet connectivity for convenience of dwellers, tourists and other clients. The cellular network is accessible via a WiFi hotspot with 3G interface. The vehicular DTN service is supported by the vehicular network installed in the municipality and bus fleets, BusNet. These vehicles are equipped with On-Board Units (OBU) that provide WiFi hotspots and vehicle-to-vehicle/infrastructure communications. To use this backhaul, the DCU opportunistically seeks WiFi hotspots advertisements from passing buses. When association occurs, data is transfered from the DCU to a service running locally in the OBU, using a CoAP [14] protocol implementation. The data is forwarded within the vehicular network to a road-side Internet gateway using a store-and-forward scheme.

The fiber ring and the cellular network have synchronous IP connectivity, i.e., an end-to-end connection through a TCP or UDP socket is established from the DCU to the backoffice. In the case of the vehicular DTN, there is no possibility of a synchronous end-to-end IP connection. The DTN implementation of service provider requires custody transfer of the data from DCUs to OBUs, and only allows unidirectional communication. This entails that delivery reliability is provided by internal mechanisms of the vehicular DTN, and that there is no possibility of sending acknowledgements from the backoffice to DCU via the DTN. Given this setup, during the data transfer from DCU to OBU, the OBU service acknowledges the received data and the DCU erases the data locally. This arrangement also motivates the use of the backoffice server as gateway to provide a unified interface for open data.

B. Application-Level Protocol

The information exchange between DCUs and backoffice server involves two types of messages: bundle messages and acknowledgement (ACK) messages. Bundle messages contain the measured data intended to be stored in the database, whereas ACK messages carry delivery confirmation. In case IP connectivity is available, the ACK comes directly from the backoffice server after writing to the central database. In case the BusNet DTN is used, the ACK is received from the backhaul hub. If the bundle or ACK is lost, another attempt to transmit the same bundle will be performed after a configurable timeout has expired. This process repeats for a configurable number of retries. If all attempts are unsuccessful, the bundle will be discarded and the corresponding data deleted. Transversely to all backhauls, local information is only removed when an acknowledgement has been received or the number of send retries is reached.

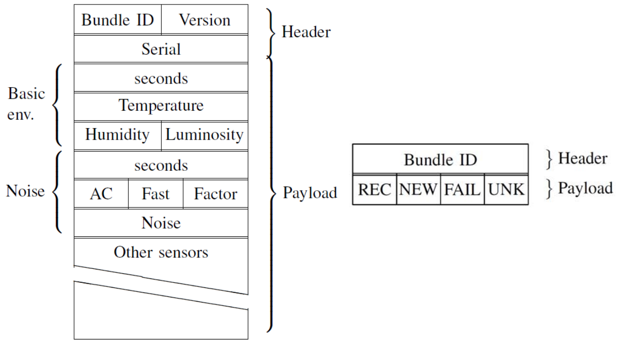
Figure 3a shows the structure of bundle messages, and Figure 3b the structure of ACK messages. The bundles messages carry a header containing a unique bundle id field and the serial number of the source DCU. The payload carries the data of the multiple sensors in a block structure – one block of fields per sensor or group of sensors, mirroring the database organization discussed in Section VI-B. Each block contains the timestamp at which the data of sensor (or sensors) was collected and the respective samples. The acknowledgement message from the backoffice server contains information about the number of sensor samples uploaded, the number of such samples successfully stored at the database, the number of unknown sensor samples types and the number of unsuccessful insertions. Note that such information is available only for end-to-end TCP connections.

VI. Software and Database Architecture

A. Software Architecture

The software architecture deployed at the processing board of DCUs has three modules:

- **Data Collector service**: in charge of polling each sensor to obtain measurement values;
- **Local database**: where sensor values are persistently stored before being sent to the central database;
- **Data Sender service**: manages data transfers to the central database at the backoffice.



(a) Bundle packets (b) ACK packets

Fig. 3. Format of the packets.

The UrbanSense server has two software components:

- **Listener service**: accepts TCP or UDP connections to receive sensor data and store it in the central database;
- **Central database**: stores all collected data for historic purposes, using a structure similar to databases in DCUs.

At DCUs, the data collector service periodically samples all the sensors and stores the sampled data in the local database. Sensor sampling is made via the control board: the service pools the control board for new data by means of a request-and-response serial protocol that supports basic error detection via checksum. Messages have been implemented to read all sensors devices and configure hardware parameters such as controlling the heating elements and power to the sonometer. The data retrieved by the data collector service requests is stored in the local database. The data sender service is in charge of sending measurement values stored in the local database to the central database. Sensor data is stored locally until acknowledged by the backoffice server (in case of synchronous connectivity) or the vehicular DTN service.

At the backoffice, the listener service performs authentication, basic data validation, stores received data in the central database and sends back an acknowledgement. The backoffice infrastructure runs on a cloud service provider where a DMZ zone has been implemented for application-layer reliability. This arrangement allows to keep the backoffice server and central database isolated from the open Internet.

B. Database Structure

The structure of the UrbanSense databases contains two classes of tables: system information/configurations, and sensor data. We briefly overview the first set of tables. The *node* table stores general information about the hardware, e.g., the RPI serial number. The *deployment* table keeps DCU deployment-specific information: location, and start and end dates. The *calibration*, *model* and *sensor* tables support calibration, accommodating evolving models via a model validity start timestamp. The table *config_settings* keeps a historical record of the settings and parameters that each DCU was operating with. It stores the name and value of each desired configuration parameter, and a timestamp indicating the instant from which each particular setting takes effect. This allows us to associate metadata and measurement parameters with collected data, e.g., the sampling rate used to collect a particular set of measurements. This design allows for flexibility in the number and variety of settings that can be enforced and to create a timeline of configuration parameters.

The tables dedicated to sensor data are organized as follows. Each entry in the sensor data is characterized by the collection time (with a precision of seconds) and the deployment that performed the measurement. It is noteworthy that there is not an individual table per sensor type. Certain sensors have been grouped together based on the fact that their samples are expected to be collected simultaneously most of the times. Each record inserted into one of these sensor data tables contains a field that is automatically filled with an insertion timestamp. This can be used to diagnose problems with the transmission of the data towards the central server or to analyze the performance of the data transport by studying the difference between acquisition and insertion times. Both central and local databases follow this model, except that calibration-related tables do not exist in the central database.

VII. Implementation Aspects

A. Time updates

The fact the Raspberry Pi lacks an on-board real-time clock (RTC) with persistence represents a challenge when it is necessary to accurately timestamp collected data. This issue is typically mitigated by using NTP to synchronize the local clock to remote time servers. In the case of DCUs with constant network connection, the clock can readily be updated over the Internet if the power is temporarily lost. In the case of DCUs using the vehicular DTN backhaul, this is not possible. To address this, the control board was designed including a RTC connected to the I^2C bus. Thus, the operating system can store date in a reliable manner and keep time accuracy, even in a disconnected scenario.

B. Sensor Calibration

We use a set of high quality sensors to develop calibration models based on linear and non-linear regressions. These equipments have higher accuracy, and are considerably more expensive and bulkier, than the platform sensors. This calibration strategy is not scalable as the procedure needs to be repeated for all locations where DCUs are deployed, and every time a sensor is installed or replaced. We address this issue by creating a set of reference DCUs that are kept in laboratory and calibrated with the high quality sensors. To perform calibration of a field DCU, a reference DCU is transported to the vicinity of a target unit to provide reference values.

To assess the feasibility of this strategy, we evaluated the accuracy of the low-cost sensors against the high quality sensors and consistency between equal-hardware sensors. Results are shown for two versions of the system, V1.1 and V2, and the dataset used has been under collection since the beginning of the project. It was found that CO concentration at measurement locations was below the sensor activation threshold; a software filter was developed, and respective results are also shown.

Table III (a) shows the correlation of measurements obtained from high quality sensors and platform sensors prior to

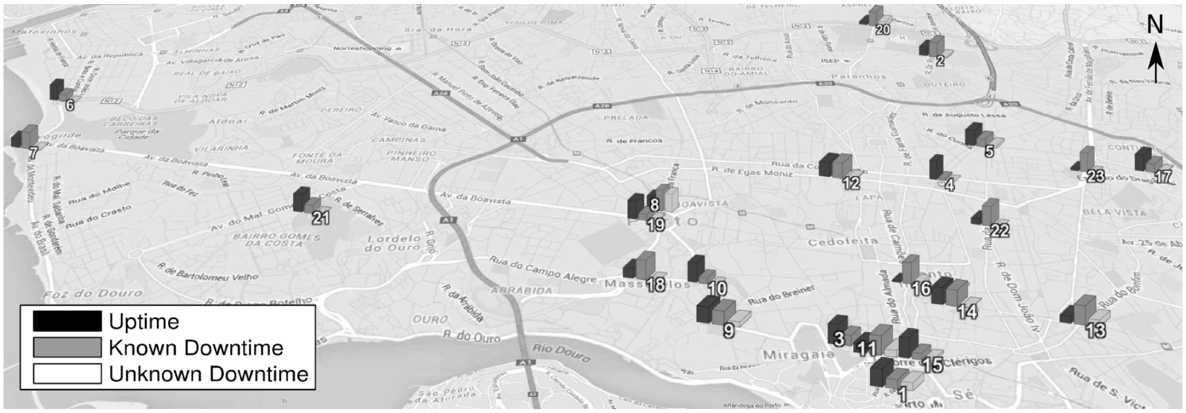


Fig. 4. Ratio of different operational status per DCU (bars per DCU sum up to 100%; numbers identify uniquely each DCU).

TABLE III
CONSISTENCY AND ACCURACY OF SENSORS

Version	O3	NO2	CO	CO (filter)	Temperat.	Humidity
V1.1	0.85	0.67	N/A	N/A	0.95	0.95
V2	0.92	0.62	0.27	0.55	0.95	0.95

(a) *Accuracy:* measurement correlation between ref. and high quality sensors.

Version	O3	NO2	CO	CO (filter)	Temperat.	Humidity
V1.1	0.98	0.97	0.02	0.06	0.97	0.97
V2	1	0.99	0.20	0.97	0.97	0.98

(b) *Consistency:* measurement correlation between similar sensor units.

calibration, to evaluate the similarity in behavior of both sets of sensors. Table III (b) shows the correlation between same-hardware sensors installed in different DCUs. The correlation is high in general, which means that variation between units is negligible. In fact, it was concluded that variation between DCUs can be modeled by a constant offset. This allows us to calibrate collected data in post-processing, using the offset between reference and field DCU.

VIII. Results and Discussion

We now evaluate two performance metrics of the platform: (i) sensing uptime ratio; and (ii) data delivery ratio. From the initial deployment of 23 DCUs, we only had sufficient data from 15 units for this analysis. Data from the remaining DCUs was not used due to technical issues, such as those discussed in Section VIII-C and communication difficulties.

A. Sensing Uptime

The sensing “uptime” ratio is the ratio between effective and expected sensor activity time. In a more detail, it is the difference between the effectively collected data and the expected amount of data that should have been collected during the test period, according to the estimates of Table II.

The expected amount of data that should have been collected is proportional to the expected activity period of the sensors. On its turn, the activity period is calculated from the total test period (6 months) minus sensor known downtime. Sensor downtime is of two kinds: known and unknown. Regarding the first, we have been carefully bookkeeping the periods at which the sensors were disconnected. Some known downtime causes are: (i) damaged sensor board; (ii) power disconnections; and (iii) SD card corruption (see Section VIII-C for more details). The second type of downtime accounts for periods for which we do not know the cause for being inoperative.

An overall sensing uptime of 54.42% was obtained. In Figure 4 we observe the three operational intervals – uptime, known downtime and unknown downtime – per Data Collection Unit. We observe no relationship between the sensing uptime, and known and unknown downtime between any of the DCUs. This helps us conclude that the factors influencing the uptime of each sensor are mostly local and not transversal to all DCUs, eliminating the possibility of a systematic error. In Figure 5, we present the sensing uptime per sensor type, for all DCUs. For most sensors, the sensing uptime is close to 100%. The sensors not achieving similar sensing uptime ratios (’Sonometer’, ’WindSpeed’, ’WindDirection’, and ’Precipitation’) share the characteristic that they are connected directly to the control board, whereas the remaining sensors are incorporated in the sensor board. Further investigation is necessary to discover if this is the cause of the inferior yields.

B. Data Delivery Ratio and Delay

Three independent communications infrastructures serve the UrbanSense platform: DTN, 3G and WiFi. We evaluate the delivery ratio and delay incurred by each of the backhauls. The number of DCUs per backhaul is the following: 7 DCUs served by DTN; 6 served by WiFi; and 2 served by 3G.

The data delivery ratio is shown in Figure 6a. We agglomerated the data of DCUs that share the same communications backbone to understand if lower data delivery ratios are associated with a particular backhaul. We observe that DCUs served by the DTN present smaller overall delivery ratio. The data delivery delay is shown in Figure 6b. It was obtained from the difference between data collection time and final database insertion time. Delay of DTN bundles scales up to hours, whereas delay of packets transported via cellular and fiber backhauls ranges between milliseconds and a few minutes. Among the later two, 3G provides the faster collection time.

C. Lessons Learned

A number of issues occurred during the test period that contribute to low sensing uptime. The main problem detected

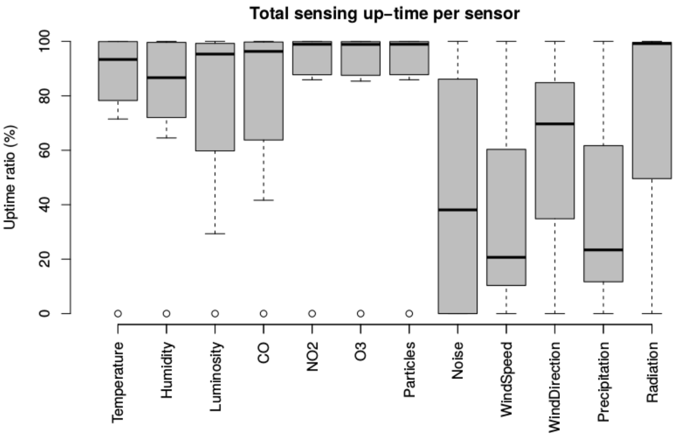
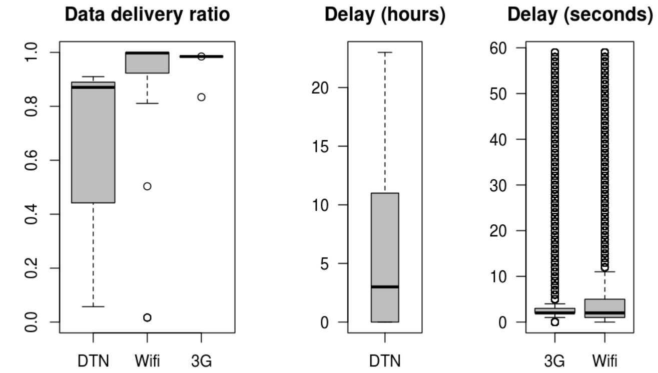


Fig. 5. Sensing uptime per sensor.



(a) Data delivery ratio.

(b) Data delivery delay.

Fig. 6. Data delivery ratio and delay boxplots for the three communication backbones. Note the different timescales for delay boxplots.

was the corruption of the local storage file system. It occurred in 10 of the 23 DCUs deployed. We are currently validating the following hypothesis to improve the sensing uptime:

- **Power outages:** we had no means to detect power outages; started to monitor and log CPU voltage.
- **Quality of SD cards:** after reviewing the current technologies used in off-the-shelf SD cards, it was decided to evaluate the performance with higher quality cards.
- **Frequency of SD card writes:** as high number of writes to the SD card may reduce its lifetime, we aim to reduce writes by keeping the database in RAM and only synchronizing at startup and shutdown.

Another issue was that sensor boards in locations close to the sea became damaged due to the adverse weather conditions. As a lesson, we learned that the use of a *watchdog* can minimize expensive human intervention at the DCUs, often for a mere hard reboot. Hence, the control board micro-controller was configured to trigger a hard reset if it does not receives any communication from the RPi after some predefined time.

IX. Conclusions

We have presented UrbanSense, a city-scale environmental sensing platform. UrbanSense is capable of monitoring multiple interest points of a city, allowing to create local and global representations of the current status and history of relevant environmental parameters. Sensor data collected at multiple sites spread around the city by sensing hubs (DCUs) is carried by an heterogeneous communications infrastructure towards a backend server where collected data is made available to interested parties via standard IoT interfaces. Currently, we are implementing a set of Key Performance Indicators to enable automatic monitoring and facilitate debugging. As future work, we plan to extend the current deployment to additional sites.

X. Acknowledgements

We thank the municipality of Porto for all the support in deploying, operating and maintaining the DCUs. We also thank Bruno Fernandes for his engagement in keeping the platform up and running, and Cecília Rocha and Sofia Sousa for providing the set of DCU deployment locations. This work was supported by the European FP7 Project Future Cities FP7-REGPOT-2012-2013-1, and by project S2MovingCity (CMUPERI/TIC/0010/2014) under the applicable financial framework (FCT/MEC through national funds and FEDER-PT2020 partnership agreement when applicable).

References

[1] J. H. Lee, M. G. Hancock, and M.-C. Hu, "Towards an effective framework for building smart cities: Lessons from seoul and san francisco," *Technological Forecasting and Social Change*, vol. 89, pp. 80–99, 2014.

[2] L. G. Anthopoulos and A. Vakali, *The Future Internet: Future Internet Assembly 2012: From Promises to Reality*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Urban Planning and Smart Cities: Interrelations and Reciprocities, pp. 178–189.

[3] J. M. Hernández-Muñoz, J. B. Vercher, L. Muñoz, J. A. Galache, M. Presser, L. A. Hernández Gómez, and J. Pettersson, *The Future Internet: Future Internet Assembly 2011: Achievements and Technological Promises*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ch. Smart Cities at the Forefront of the Future Internet, pp. 447–462.

[4] 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.

[5] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh, "Citysense: An urban-scale wireless sensor network and testbed," in *Technologies for Homeland Security, 2008 IEEE Conference on*, May 2008, pp. 583–588.

[6] R. Sombrutzki, A. Zubow, M. Kurth, and J. Redlich, "Self-organization in community mesh networks: The berlin roofnet," in *1st Workshop on Operator-Assisted Community Networks*, Sept 2006, pp. 1–11.

[7] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: a wireless sensor network testbed," in *IPSN 2005. 4th International Symposium on Information Processing in Sensor Networks*, April 2005, pp. 483–488.

[8] M. Mead, O. Popoola, G. Stewart, P. Landshoff, M. Calleja, M. Hayes, J. Baldovi, M. McLeod, T. Hodgson, J. Dicks, A. Lewis, J. Cohen, R. Baron, J. Saffell, and R. Jones, "The use of electrochemical sensors for monitoring urban air quality in low-cost, high-density networks," *Atmospheric Environment*, vol. 70, pp. 186 – 203, 2013.

[9] MIT, "Clairity - an air quality sensor netw for mit's campus," http://clairity.mit.edu/site/documents/report.pdf.

[10] J. J. Li, B. Faltings, O. Saukh, D. Hasenfratz, and J. Beutel, "Sensing the air we breathe: The opensense zurich dataset," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, ser. AAAI'12. AAAI Press, 2012, pp. 323–325.

[11] Acronet, "Acronet paradigm open hardware monitoring systems," Online, 2016, http://www.acronet.cc/.

[12] Fiware Community, "Fiware," Online, 2016, http://www.fiware.org/.

[13] ESTI, "ETSI TS 102 690 V2.1.1 (2013-10) Machine-to-Machine communications (M2M); Functional Architecture," 2013.

[14] K. H. Z. Shelby and C. Bormann, *The Constrained Application Protocol (CoAP), RFC 7252*, Internet Engineering Task Force (IETF), https://tools.ietf.org/html/rfc7252/, June 2014.