Towards a Novel Engine to Underlie the Data Transmission of Social Urban Sensing Applications

Carlos Oberdan Rolim[1], Anubis Graciela de Moraes Rossetto[1], Valderi R. Q. Leithardt[1], Guilherme A. Borges[1], Tatiana F. M. dos Santos[2], Adriano M. Souza[2] and Claudio Geyer[1]

[1]*Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil*

[2]*Postgraduate Program in Production Engineering, Federal University of Santa Maria (UFSM), Santa Maria, Brazil*

Keywords: Urban Sensing, Smart Cities, Opportunistic Networks, Machine Learning, Prediction, Situation Awareness.

Abstract: Social urban sensing is a new paradigm which exploits human-carried or vehicle-mounted sensors to ubiquitously collect data for large-scale urban sensing. A challenge of such scenario is how to transmit sensed data in situations where the networking infrastructure is intermittent or unavailable. In this context, this paper outlines the early stages of our research which is concerned with a novel engine that uses Opportunistic Networks paradigm to underlie the data transmission of social urban sensing applications. It applies Situation awareness, Neural Networks and Fuzzy Logic for routing and decision-making process. As we know, this is the first paper to use such approaches in Smart Cities area with focus on social sensing application. As well as being original, the preliminary results from our simulations signals the way that further research can be carried out in this area.

1 INTRODUCTION

Smart Cities are urban systems that use Information and Communication Technologies (ICT) to provide an infrastructure and public services within a more interactive, accessible and efficient city (Pellicer et al., 2013). As a result, researchers are seeking alternatives to serve citizens with new services to improve their quality of life and to fulfill the criteria of energy efficiency and sustainability. In this way, urban sensing applications emerges as a promising way to improve the comprehension of such urban ecosystems in order to assist the decision-makers in the organization of the city and the welfare of its residents.

In this context, an important question is how to foster citizen participation and community involvement to achieve a better interaction with the urban ecosystem. Among these initiatives, social urban sensing applications are a promising way to promote the sensing of data about different aspects of cities, bringing the computational world and community closer together.

A challenge for social urban sensing applications is how to transmit sensed data in situations where the networking infrastructure is intermittent or unavailable. We argue that Opportunistic Networks is an alternative to overcoming such limitations. Opportunistic Network is a recent and promising mobile networking paradigm that stem from research into conventional Mobile Ad Hoc NET-works (MANET) and uses contact between mobile nodes to transmit data (Boldrini et al., 2010).

In this paper, we outline the early stages of our research on a novel engine that uses Opportunistic Networks paradigm to underlie the data transmission of social urban sensing applications. It applies Situation awareness, Fuzzy Logic and Neural Networks for routing and decision-making process. This engine will be used as an internal Communication component in our Ubiquitous Service-Oriented Architecture for Urban Sensing called UrboSenti (Rolim et al., 2014). We are based on the hypothesis that a non "IP-Centric" paradigm like Opportunistic Networks in conjunction with contextual and intelligent approaches could supply the requirements of data transmission in wide-scale urban scenarios. In summary, the main contributions made by this paper are the architecture of the engine and the conceptual models used as guidance for its development. As we know, this is the first paper to use such approaches in Smart Cities area with focus on social sensing application.

The rest of this paper is structured as follows: The

DOI: 10.5220/0005457406620667
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 662-667
ISBN: 978-989-758-097-0

next section describes the motivational scenario and raises some of the current computational challenges; Section 3 describes the proposed architecture; Section 4 presents preliminary results from simulations, and, finally, in Section 5 some conclusions are reached, together with recommendations for future research.

2 PROBLEM SCENARIO

Our research has been driven by the problem-scenario that is shown in Fig. 1. This scenario includes a city with several data sources that are being used for sensing. Human-carried, fixed or vehicle-mounted sensors are applied for obtaining sensing maps of transits, air quality, noise levels, temperature, CO2 concentration, etc. Moreover, data from social networks in conjunction with sensors data are crucial to understand the behavior of the city and to provide a holistic view about it. To collect, analyze and give feedback of sensed data acquired from several sources scattered along the city, we are using our Ubiquitous Service-Oriented Architecture for Urban Sensing called *UrboSenti*. The main function of UrboSenti is to provide support for overall process of urban sensing. It splits into two key modules: the *Backend module* and *Sensing module*.

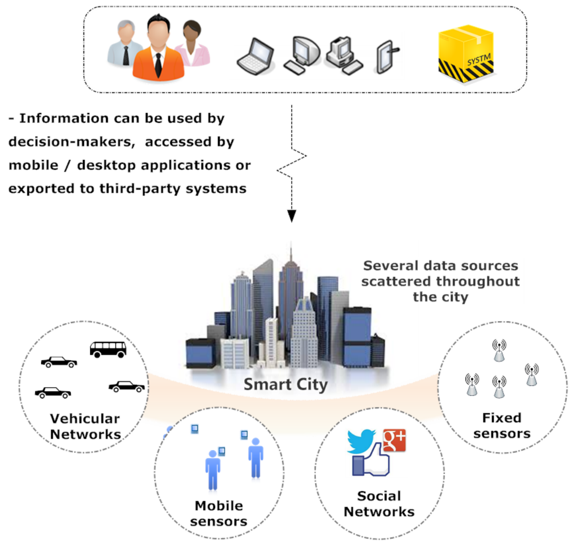

- Information can be used by decision-makers, accessed by mobile / desktop applications or exported to third-party systems
SYSTM
Several data sources scattered throughout the city
Smart City
Vehicular Networks
Fixed sensors
Mobile sensors
Social Networks

Figure 1: Problem scenario.

The *Backend module* runs in a data center infrastructure and, in short, is responsible for receiving sensed data, processing it and giving feedback to the citizens and other systems.

The *Sensing module* is responsible for social and traditional sensing and encompasses activities of intentional and non-intentional sensing. It runs in mobile devices (e.g. mobile phones, embedded in vehicles, etc) and in fixed sensors scattered around the city. It has a several components that could be plugged “on demand” and a micro-kernel with set of components that are responsible for essential features. Our focus is the internal micro-kernel *Communication* component. It provides methods to send and receive data by means of the available network infrastructure, such as IEEE 802.11b/g/n (structured and ad-hoc), GPRS/EDGE/3G and Ethernet as the underlying system for TCP/UDP communications. When the network infra-structure is intermittent or unavailable, it supports data communication using alternative ways, not based in end-to-end paths (like used in IP communications). In summary, the *Communication* component is the “power-horse” of all communications tasks in the *Sensing module*.

Hence, we are seeking for an engine to be used as underlying for communication of such component. It should handle with fuzziness of wide-scale urban scenarios and provide suitable support for data transmission of urban sensing applications running atop of UrboSenti. For such task, the requirements are: (i) to use non “IP-Centric” paradigm for communication; (ii) adapts itself the transmission parameters according to device used for sensing and the current context; (iii) made adaptation decisions proactively; (iv) perform decisions using uncertain data; (v) concerns with processing and power restrictions of devices.

In this way, we are based on the hypothesis that non “IP-Centric” paradigm like Opportunistic Networks could be used for data carrying, satisfying requisite (i); Situation awareness could be applied to deal with context adaptations, satisfying requisite (ii); Neural Networks could make predictions to support adaptations, satisfying requisite (iii) and; Fuzzy Logic could be used for decision-making about routing and internal adjusts, satisfying requisite (iv). We highlight that all approaches are suitable to run in low powered devices, satisfying requisite (v).

3 PROPOSED ENGINE

A challenge for social urban sensing applications is how to transmit sensed data in situations where the networking infrastructure is intermittent or unavailable. We argue that Opportunistic Networks paradigm could fulfill such gap. There are some initiatives in Opportunistic Networks are

Our proposed engine is lying in Opportunistic Networks area. There are some initiatives in such area, like Epidemic, Spray&Wait (and the Spray variants like Spray&Focus, Fuzzy-Spray and others), Prophet, BubbleRap, MobySpace, AFRON, Cartoon, CAR, HiBOp, CiPRO, Propicman and the most recent Prophet improvement called DRAFT. Due to space limitation and paper scope we will not present more information about these initiatives – for further information see (Jedari et al., 2013). However, we remark that none could handle all requirements presented above and cannot be used "as is" in our scenario. Thus, in this section we will present our engine architecture to overcome such limitations.

3.1 Situation Awareness Model

Situation-awareness (SAW) is a computing paradigm by which applications use context data to sense and comprehend the current situation and to project the future demands. The first formalization of SAW was a 3-levels model proposed by Endsley(Endsley, 1995). To deal with context adaptation in a proactive fashion, our engine implements a Situation awareness model. It is based in 3-levels model proposed by Endsley and is outlined in Fig. 2.

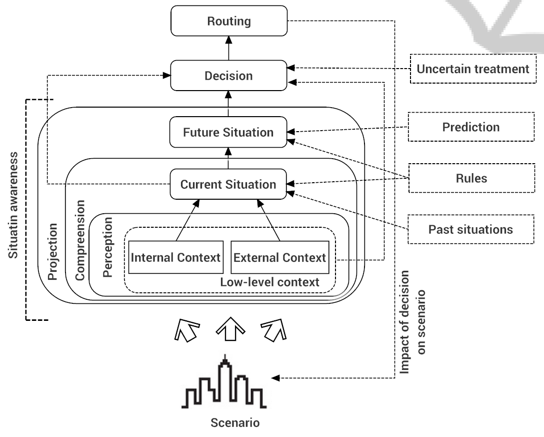


Figure 2: Situation model.

This model uses internal and external context about node to derive the low-level context that characterizes current situation. A set of rules and past situations are used to project future situations that will be used for routing decisions.

3.2 Prediction Model

The Prediction model (Fig. 3) is used to project future situations. A recurrent Neural Network (NN) is used for this purpose. We have chosen NN due to its capacity to solve non-linear problems, it are universal functions aproximators being suitable for prediction.

Each node is responsible to train and runs its own instance of NN. With this approach we ensure that

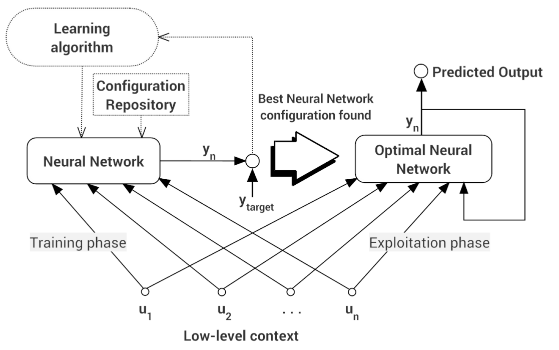


Figure 3: Prediction model.

each node have a suitable NN to its needs. The current and past low-level context data are used as input for NN. It starts the Training phase, testing several configurations from a configuration repository trying to find the optimal Network (with lower Root Mean Squared Error – RMSE). When a optimal Network is found, it is used for prediction in Exploitation phase.

The outputs of this process are new predicted low-level context data that probably characterize a future situation of node. These data are used in decision-making process. An essential requirement for NN in this case, is a low computational cost due to power and processing constraints of mobile nodes.

3.3 Decision Making Model

The decision-making process encompasses precise and non precise context data (from current and projected future situation). So, conventional logic may produce completely wrong decisions due to uncertain of such data. An alternative to deal with imprecise data is Fuzzy Logic. According to (Zadeh, 2008), Fuzzy Logic is a viable alternative to reason and make rational decisions in an environment of imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and partiality of possibility. Thus, to perform decision-making process, a Fuzzy Inference System (FIS) is used (Fig. 4). Context values (current and future) are pumped as input to FIS and after internal calculation (by usage of membership functions and fuzzy rules) the output is the potential of a node to be a good "data mule" (a term used in Opportunistic Network to designate a node that will carry a message). This information will be used to decide when to forward messages to encountered node. The decision-making process also runs to decide if some internal parameters needs to be adjusted.

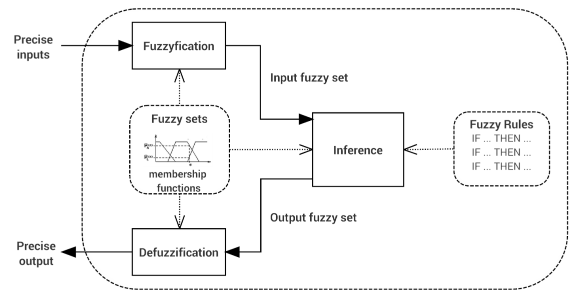


Figure 4: Decision making model.

3.4 Engine Architecture

The above presented models are used by our engine. Its internal architecture features are outlined in Fig. 5 and its behavior is explained below. Again, we highlight that due to space limitation we will not fully explain each component.

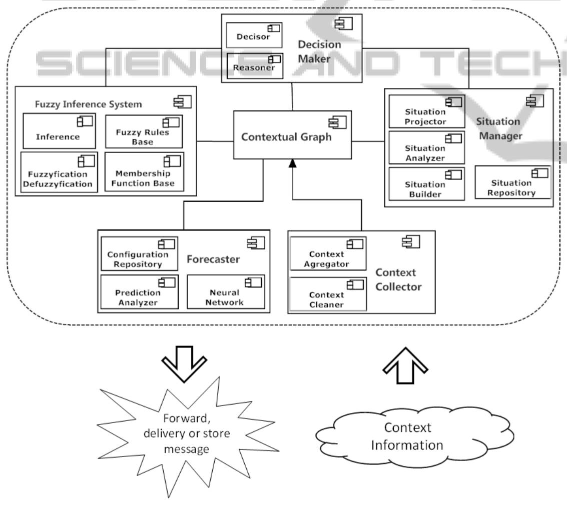


Figure 5: Engine architecture.

It starts with the *Contextual Information* that representing information about the context of node. At a constant time interval, *Context Collector* collects the data and stores them in a layered structure called *Contextual Graph*, thus creating a new Layer 2 vertex.

The *Contextual Graph* component underlies all the data storage. Its main function is to store instantaneous and predicted context information. In Contextual Graph, the vertexes of the graph are structured in layers: Layer 1 stores basic information about the node (i.e. node name, address, network interfaces). Layer 2 stores instantaneous context information about the node (i.e. node power, current position, buffer usage, number of messages, current time, speed, distance traveled from last point, number of reachable neighbors) that will be used as historical values to prime the Forecaster. Layer 3 stores predicted context values from Forecaster. Moreover, we used edges to represent the contacts between the nodes.

The *Situation Manager* component, implements our Situation awareness model. It draws on data from *Contextual Graph* to build, analyze, project and create a repository of situations. The information generated by this module will be used later by the *Decision Maker*. It also runs maintenance routines like pruning old data and invoking *Forecaster* for prediction. At time intervals *Situation Manager* retrieve context data from *Contextual Graph* and using a set of rules stored in its internal situation repository it tries to identify (build) the current node's situation. The identified situation (e.g. "node is sensing with low battery power and high buffer usage") is analyzed, and if it indicates that some action needs to be done it is reported to *Decision Maker*. If a situation could not be identified, an unknown situation is found. Thus, a new set of rules that characterize this situation is created "on the fly" and stored in repository for future use. When the *Situation Manager* component detects a sufficient amount of context information, it could project a future situation. For this purpose, the *Forecaster* component is invoked.

Forecaster component implements our prediction model, in order to predict the probable values that will characterize a future situation. For such task, we applied a lightweight network called Echo State Network (ESN) proposed by Jaeger(Jaeger, 2001) is applied. ESN are a kind of three-layered recurrent network with sparse, random, and (crucially), untrained connections within the recurrent hidden layer (for further and mathematical foundation please read original Jaegers paper). The main difference between the ESN and other neural networks is that only the weights of the reservoir output signals are trained. The weights of the connections within the reservoir are not trained but are randomly generated. This approach significantly reduces the learning process when compared with other algorithms (e.g. back propagation through time) resulting in low computational cost to implement it(Fink et al., 2013). The Forecaster component, uses context values stored as Layer 2 in *Contextual Graph* as historical data to train ESN (i.e. node power, current position, etc.). Due to its low computation cost to train the neural network, we are able to make each node of the network to builds its own ESN with the most appropriate configuration for its context. This is carried out by testing different internal parameters of ESN (i.e. size of reservoir, sparsity of the reservoir, spectral radius and leaking rate) with different values until the best one (i.e. the one with minimal MSE) is found. When optimal neural network is found, its configuration is stored. At this point, optimal network is ready to predict future values in the exploitation phase. During exploitation phase, the structure with historical data is used to "pump" the best network (found and saved in previous phase) with some data steps and thus to activate the internal reservoir. Some stages later, the input from the historical data is switched off to allow the network to predict values by itself. The predicted values are stored at *Contextual Graph* as Layer 3 vertices. At this point, the component sets an internal variable to indicate that the node is now running in smart mode. In smart mode, all decisions done by *Decision Maker* are made using past, current and predicted data in order to improve data transmission. In "dummy" mode, just current situation is used in decisions.

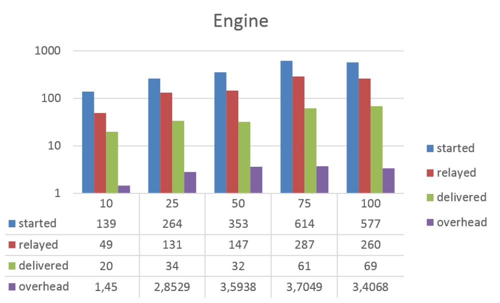
Decision Maker runs at constant time intervals to decide if some internal parameters needs to be adjusted (such as buffer scheduling policy, maximum size of messages, time to live of new messages, etc.) or if a "trap" should be triggered to require attention of an external component of micro-kernel (e.g to change configuration of network interface, to perform some adaptation action, etc.). All decisions are made using decision making model presented above in section 3.3. *Decision Maker* is also invoked when current node contact another node to decide if some buffered message should be forwarded, delivered or remain at the local buffer. In simple terms, *Decision Maker* decides if the encountered node is a good "data mule". We used the term "potential" to represent the capacity of the node to be a good data mule. The strategy used is quite simple: if the potential of the contacted node is greater than the potential of the current node, then the message is forwarded; otherwise, the message remains at the local buffer (obviously the message is delivered if the encountered node is its destination). The question arising from this approach is: how to calculate the potential of each nodes? For this task, all context values (current, historical and predicted) of the current and contacted node from *Contextual Graph* and use it as input for the *Fuzzy Inference System (FIS)*. *FIS* uses its internal components and rules to calculate the potential of each node. The potential values of nodes are used by *Decisor* in decision making process.

4 PRELIMINARY RESULTS

To verify the functionality and performance of proposed engine we implemented some main modules and carried out some simulations using ONE (Opportunistic Network Environment) Simulator. For simulation setup we adopted 6 hours for all scenarios with a different number of nodes (10 for the first, 25 for the second, and 50, 75 and 100 for each consecutive group). We used two groups: pedestrians and cars with ShortestPathMapBasedMovement as mobility model. Pedestrian nodes moved between 0.5 and 1.5 Km/h, and had a Bluetooth device with a radio range of 20 meters and transmission speed of 2 Mbit/s. The Car nodes moved between 10 and 50 Km/h and had a Wi-Fi interface with a range of 50 meters and transmission speed of 10 Mbit/s. On average, the nodes generated about one message every 25 to 35 seconds (total of 711) and the message lifetime was set at 24 minutes (1440 seconds). We used message sizes that were uniformly distributed between 100 KB and 2 MB.

In implementation of engine, we used ESN-Java software[1] to build Forecaster and JFuzzyLogic library[2] in the Fuzzy Inference System used by Decision-Maker. The following context data was used: current power, current speed, total distance traveled from last point, overall distance traveled, current coordinates, last coordinates, current buffer usage, current number of carried messages, total number of forwarded messages, current number of neighboring nodes, and total number of connections. To calculate the variable "potential" which is used as the output of FIS, three Triangular membership function are used. The COG (Center Of Gravity) was used as a defuzzification method. The fuzzy inference rules were defined in compliance with Fuzzy control language (FCL).

We conducted a set of experiments using simulation setup presented above with different scenarios. The results are displayed in Fig. 6. This shows that in general, there is an increment in the number of delivered messages that corresponds to the increment of the number of nodes.



	10	25	50	75	100
■ started	139	264	353	614	577
■ relayed	49	131	147	287	260
■ delivered	20	34	32	61	69
■ overhead	1,45	2,8529	3,5938	3,7049	3,4068

Figure 6: Performance in scenarios with different number of nodes.

[1]http://www.wsi.uni-tuebingen.de/lehrstuehle/cognitive-modeling/code/overview.html

[2]http://jfuzzylogic.sourceforge.net/

The scenario with 100 nodes had a increment of 245% in number of delivered messages in relation to scenario with 10 nodes, but with just 134% of more overhead (an assessment of bandwidth efficiency in relation of the number of relayed and delivered messages). This overhead percentage is less than in scenario with 50 and 75 nodes. We believe that this ratio can be attributed to our strategy of just relaying messages to data mules with a good potential to delivery the message. Furthermore, we can note that with increment of number of nodes, each one could store more historical context data to be used by Forecaster for prediction. With this, the predictions becomes more accurate over the time. In other words, the engine becomes smarter when run more time and with more neighbors nodes. We need to investigate why with 50 nodes the number of delivered messages was less than with 25 nodes. I As final remark, one factor not reported in the chart is the computational cost of ESN. Even when each node used in the simulation testing, had, on average, 810 different configurations to find the best network, the impact of the processor load was minimal. This lightweight feature was the main differential of ESN when compared with all the other machine learning approaches that we have tested in our previous work.

5 CONCLUSIONS

In this paper, we have described the early stages of our attempt to build a novel engine that applies Opportunistic Networks paradigm to transmit sensed data in situations where the networking infrastructure is intermittent or unavailable. It runs as an internal component of a wide architecture called UrboSenti and provides support for communication of urban sensing applications running atop of it.

We have also outlined our initial design models for the software modules and their internal components. The development of engine has been started. Currently we are mainly working to implement Situation awareness model and to plug it with other components. The preliminary results, without situation awareness, are acceptable and indicates that our initial hypothesis could be exploited better. The low computational cost to run it with satisfactory number of delivered messages has shown that it works and have a good potential to be used in UrboSenti. We believe that its performance will be improved when implementation of Situation Manager is finished.

Thus, we claim that the proposed engine attend our requirements and is able to fill the gap of data transmission presented in our initial problem-scenario. Moreover, this should encourage us to conduct further research into the multidisciplinary area of Smart Cities with the aim of improving services and applications for urban sensing.

For future work, we are seeking alternative means of constructing fuzzy sets and rules "on the fly", depending on the situation in which the node is immersed and to explore the application of a Deep Belief Network (DBN) or Restricted Boltzmann machines (RBMs) as underlying for prediction.

ACKNOWLEDGEMENTS

We would like to give special thanks to Johannes Lohmann of Universität Würzburg for his help with source code of ESNJava.
This research was supported under UbiArch project – Ubiquitous Architecture for Context Management and Application Development at UFRGS.

REFERENCES

Boldrini, C., Conti, M., and Passarella, A. (2010). Design and performance evaluation of ContentPlace, a social-aware data dissemination system for opportunistic networks. *Computer Networks*, 54(4):589–604.

Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems.

Fink, O., Zio, E., and Weidmann, U. (2013). Anticipating Railway Operation Disruption Events Based on the Analysis of Discrete-Event Diagnostic Data. *Ehemical Engineering Transactions*, 33:715–720.

Jaeger, H. (2001). The echo state approach to analysing and training recurrent neural networks. Technical report GMD report 148. Technical report, German National Research Center for Information Technology.

Jedari, B., Xia, F., and Member, S. (2013). A Survey on Routing and Data Dissemination in Opportunistic Mobile Social Networks. *IEEE Communications Surveys and Tutorials*.

Pellicer, S., Santa, G., Bleda, A. L., Maestre, R., Jara, A. J., and Skarmeta, A. G. (2013). A Global Perspective of Smart Cities: A Survey. *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 439–444.

Rolim, C., Rossetto, A., Leithardt, V. R. Q., Borges, G., dos Santos, T. F. M., Souza, A., and Geyer, C. F. R. (2014). Towards a Ubiquitous Service-Oriented Architecture for Urban Sensing. In *ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference*. Academy of Science and Engineering (ASE), USA, ASE 2014.

Zadeh, L. a. (2008). Is there a need for fuzzy logic? *Information Sciences*, 178(13):2751–2779.