Putting Opportunistic, Situational and Smart Approaches to Underlie the Data Transmission of Social Urban Sensing Applications

Carlos O. Rolim[1], **Anubis G. Rossetto**[1], **Valderi R. Q. Leithardt**[1]
Guilherme A. Borges[1], **Tatiana F. M. dos Santos**[2]
Adriano M. Souza[2], **Claudio F. R. Geyer**[1]

[1]Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS)
91.501-970 – Porto Alegre – RS – Brazil

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

carlos.oberdan@inf.ufrgs.br, taty.nanda@gmail.com

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 our researches on an engine that uses Opportunistic Networks paradigm to underlie the data transmission of social urban sensing applications. It also applies Situation awareness, Fuzzy Logic and Neural Networks to perform routing, adaptation and decision-making process. We carried out simulations using a simulator environment, achieving positive results. As we know, this is the first paper to use such approaches in Smart Cities area with focus on social sensing application.*

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]. In such context, 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 "feel the pulse" of the city, improving 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.

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.

In this paper, we outline our research on an engine that uses Opportunistic Networks paradigm to underlie the data transmission of social urban sensing applications. Moreover, it also applies Situation awareness, Fuzzy Logic and Neural Networks to perform routing, adaptation and decision-making process. This engine will be used as a internal Communication component in our Ubiquitous Service-Oriented Architecture for Urban Sensing called UrboSenti [Rolim et al. 2015].

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. As well as being original, signals the way that further research can be carried out in this area.

The rest of this paper is structured as follows: The next section describes the motivational scenario and raises some of the current computational challenges; Section 3 provides a brief outline of some background concepts and related initiatives; Section 4 describes the proposed architecture; Section 5 describes our experiments and analyzes the results; and, finally, in Section 6 some conclusions are reached, together with recommendations for future research.

2. PROBLEM SCENARIO

Our research has been driven by the problem-scenario of 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, CO_2 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*.

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.

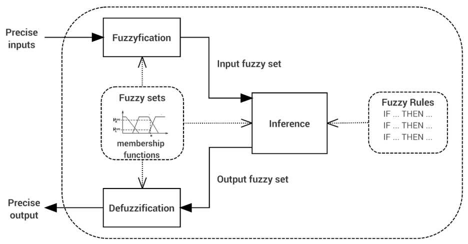
The *Sensing module* is responsible for social and traditional sensing and encompasses activities of intentional and non-intentional sensing. It runs in mobile devices (e.g. mobile phones, embedded in vehicles, etc) and in fixed sensors scattered around the city. It has a several components that could be plugged "on demand" and a micro-kernel with a set of components that are responsible for essential features. Our focus is the internal micro-kernel component namely *Communication*. 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). The selection of best method for transmission is made by engine itself, without interaction of user. In summary, the *Communication* component is the "power-horse" of all communications tasks in the *Sensing module*. Hence, we are seeking for a self-adaptive engine to be used as underlying for such component with following requirements: (i) use a non "IP-Centric" paradigm for communication; (ii) provide support for buffer management; (iii) adapts itself the transmission parameters according to device used for sensing and the current context; (iv) made adaptation decisions proactively; (v) concerns with processing and power restrictions of devices.

In this way, we are based on the hypothesis that Opportunistic, Situational and Smart Approaches could fulfill such requisites. For "Opportunistic" we are talking about Opportunistic Networks, a non "IP-Centric" paradigm, that could be used for data carrying, satisfying requisite (i); For "Situational" we are referencing to Situation awareness that could be applied to deal with context adaptations, satisfying requisite (ii); For "Smart" we are talking about machine learning approaches as Fuzzy Logic and Neural Networks. Neural Networks could make predictions to support adaptations proactively, 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).

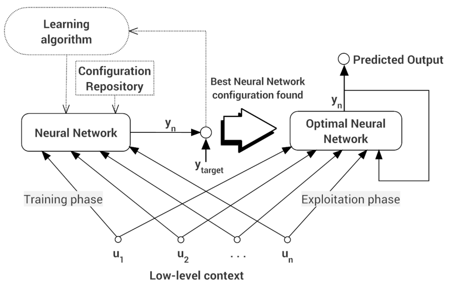
3. Background and related works

Opportunistic Networks is a new emergent network paradigm. It seeks to simplify the complexity at the network layer by removing the assumption of physical end-to-end connectivity while providing connectivity opportunities for mobile devices when networking infrastructure is intermittent or unavailable. In Opportunistic Networks, the forwarding of messages is based on the Store-Carry and Forward concept. In such way, different initiatives have been employed to route messages from their source to their destination in a suitable way, since end-to-end paths might be absent for the whole life-time of the message. We could cite: 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]. We argue that none of then could fulfill the requirements present in section 2 and cannot be used "as is" in our scenario. We have used some concepts of context information for routing decisions like Cartoon, but instead of just relying on instantaneous information, we have base our decision-making on the probable future situation. Moreover, our work is sited in the same Artificial Intelligence area of CAR protocol. However instead of depending on Kalman Filters for prediction and multi-criteria decision theory when choosing the best next hop for the message, in our work we have applied fuzzy logic for decision making and neural networks as underlying technique for prediction.

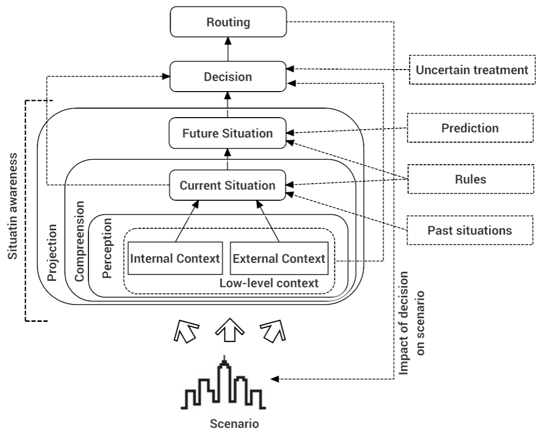
Echo State Networks (ESN)[Jaeger 2001] are a kind of three-layered recurrent network with sparse, random, and (crucially), untrained connections within the recurrent hidden layer. As seminal paper makes clear, ESN has the prospect of offering significant performance benefits. The main structural element of ESN is a reservoir rather than a layered structure. The weights between the connected neurons within the reservoir are fixed and are not trained during the training process 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. Such appointment motivated us to use ESN as underlying technique for forecasting. ESN have been applied to solve practical problems in various domains. With regard to forecasting by means of ESN, the following can be cited [Yu et al. 2011] [Rabin et al. 2013]. At this time, we cannot find any work that applies ESN in social urban sensing applications.



(a) Decision-making model



(b) Prediction model



(c) Situation model

Figura 1. Conceptual models

The fuzzy set theory was proposed by Zadeh in 1965 as an extension of multivalued logic. It has been described as a precise logic of imprecision and approximate reasoning. In the area of ubiquitous computing, it has been used for different purposes, like location prediction[Anagnostopoulos et al. 2011] and context inference[Perttunen et al. 2009]. We are using fuzzy logic because it is able to support real-time decisions in situations where there is some degree of uncertainty and vagueness with regard to the context determination. We cannot find any work that uses Fuzzy Logic in social urban sensing area. The nearest initiatives found was in Opportunistic Networks field. Fuzzy Logic already was used by AFRON to find the destination node in shortest time with less buffer usage and Adaptive Fuzzy Spray and Wait that is concentrated on optimizing the number of message copies to be disseminated in the network. However, none of these works provide support for all of our requirements.

4. PROPOSED SOLUTION

4.1. Conceptual models

The conceptual models used in our engine are depicted in Fig. 3. The **Decision-making model** (Fig. 1a) deals with decisions of routing and about adaptation of our engine. It 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 handle imprecise data is Fuzzy Logic. 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.

The **Prediction model** (Fig. 1b) 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 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.

The **Situation awareness model** (Fig. 1c) is used to deal with context adaptation in a proactive fashion. It is based in 3-tier model proposed by [Endsley 1995]. 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.

4.2. Engine architecture

The above presented models are used by our engine. Its internal architecture features are outlined in Fig. 2 and its behavior is explained below.

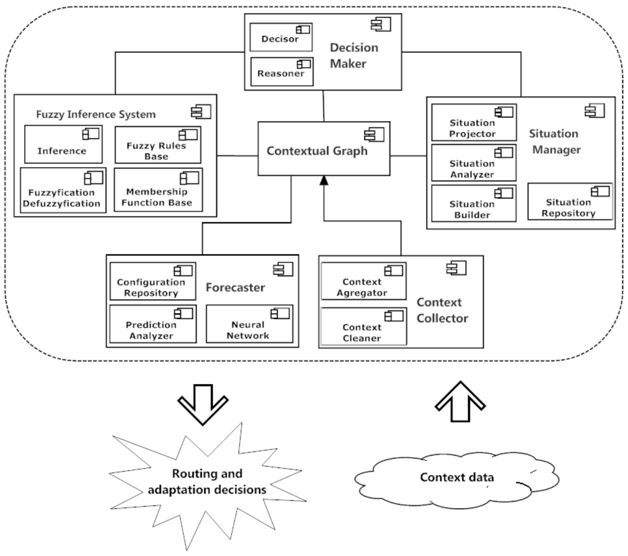


Figura 2. Engine architecture

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. Neo4j[1] graph database was used as the underlying mechanism for this purpose. 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, etc.) 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, it 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 in a self-recurrent way. 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 implements our Decision-making model. It 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

[1]http://www.neo4j.org/

"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.). *Decision Maker* is also invoke 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, past and predicted) of the current and contacted node from *Contextual Graph* are used as input for the *Fuzzy Inference System (FIS). FIS* uses its internal components and rules to calculate the potential of each node.

5. SIMULATION AND EXPERIMENTAL RESULTS

5.1. Simulation setup

To verify the functionality and performance of proposed engine we implemented the 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 of sensing data 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.

The ESN used in *Forecaster* was built using ESNJava software[2]. As each node executes their own neural network, resulting in specific configuration, we have not shown configuration results for each node, but only the results from the average of all the nodes. On average, around 104 steps were used in the training phase and 54 steps in the exploitation phase. The MSE in the training phase ranged from 4.01e-08 to 4.55e-10 and in the exploitation phase ranged from 1.21e-7 to 7.67e-8. The internal size of the network ranged from 10 to 20 nodes and the spectral radius from 0.77 to 0.85. The only value that was fixed for all the nodes was the sparsity of the reservoir = 1 and leaking rate = 0.

The Fuzzy Inference System (FIS) used in *Decision Maker* was implemented using JFuzzyLogic library[3]. 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,

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

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

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 highlight that currently we are working in Situation Manager. So, the following experiments was done just with a basic situation set, without incorporating new situations in repository.

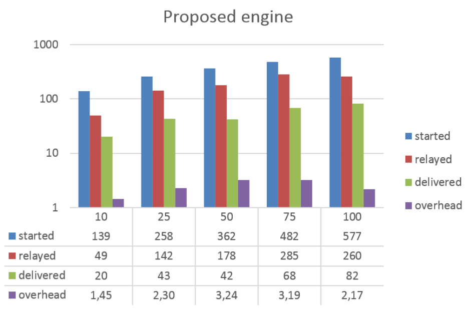
5.2. Experimental results

We conducted a set of experiments using simulation setup presented above with different scenarios. The results are displayed in Fig. 3a. 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.

The scenario with 100 nodes had a increment of 310% in number of delivered messages in relation to scenario with 10 nodes, but with just 49% of more overhead (an assessment of bandwidth efficiency in relation of the number of relayed and delivered messages). We can note that 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. However, it could be improved with tunning of strategy used in buffer management. 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 longer and with more neighbors nodes. We need to investigate why with 50 nodes the number of delivered messages was less than with 25 nodes. One factor not reported in the chart is the computational cost of ESN. Even when each node used in the simulation testing several different configurations to find the best network, the impact of the processor load was minimal. This lightweight feature was the main finding of application of ESN in our engine.

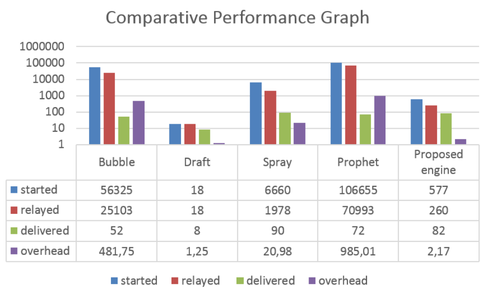
We also compared our engine with some approaches used “as is” from Opportunistic Networks area. We used the same scenario setup of previous experiments. The protocols used for comparison were Prophet, DRAFT, Spray&Wait and BubbleRap. These were chosen because they represent the different classes of “intelligent” protocols and have been extensively studied by researchers[Orlinski and Filer 2013]. We used the following configuration parameters: for Prophet we set secondsInTimeUnit = 30. For DRAFT we set familiarThreshold = 120, degrade = 0.5 and frame size = 3600. For Bubblerap we set K = 5, familiarThreshold = 700, centralityTimeWindow = 3600 and epoch_count = 6 (i.e. simulation time of 21600 seconds centralityTimeWindow 3600 = 6). The comparison results of selected approaches in a scenario with 100 nodes are presented in Fig. 3b

As can be seen, Spray&Wait and Prophet have results similar to our engine in number of delivered messages. Our approach is just worse than Spray&Wait. However, we highlight that in the number of started and relayed messages and the overhead ratio of bandwidth we are almost 12% better. This result comes from low number of startedrelayed messages and witness that our strategy to just relay messages to better data mules is saving computational resources. We recall that to relay messages, mobile de-



	10	25	50	75	100
■ started	139	258	362	482	577
■ relayed	49	142	178	285	260
■ delivered	20	43	42	68	82
■ overhead	1,45	2,30	3,24	3,19	2,17

(a) Performance in scenarios with different number of nodes



	Bubble	Draft	Spray	Prophet	Proposed engine
■ started	56325	18	6660	106655	577
■ relayed	25103	18	1978	70993	260
■ delivered	52	8	90	72	82
■ overhead	481,75	1,25	20,98	985,01	2,17

(b) Comparison with Opportunistic Networks approaches

Figura 3. Comparison performance

vices use battery power for processing and data transmission; thus, a high value of this overhead indicator could influence the battery life of these devices. Draft had the lowest number of delivered messages. Despite of acceptable number of delivered messages of BubbleRap and Prophet the generated overhead was huge in comparison with our engine. These initiatives have a good performance in some Opportunistic Networks scenarios, but as we suspected when it was used in social sensing scenario they are not suitable.

Thus, the results indicates that our engine had a satisfactory performance in terms of the number of delivered messages and overhead ratio and could be used in wide-scale urban scenarios where network infrastructure is intermittent or unavailable, such as Smart Cities.

6. Conclusion and Future work

In this paper, we have described our attempt to build an engine that applies Opportunistic Networks paradigm to transmit sensed data in situations where the networking infrastructure is intermittent or unavailable. It runs in 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. Currently we are mainly working to implement Situation awareness model. The preliminary results, with a basic situation set, are acceptable. 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 done. The experiments also made clear that ESN is a good technique for prediction. It achieved an impressive predictive performance and has a low computational cost compared with all the other approaches that we already applied. In addition, the results depicted that some popular Opportunistic Networks initiatives could not be used "as is" in urban sensing applications area.

As final remark, we claim that the proposed engine 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.

7. Acknowledgments

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.

Referências

Anagnostopoulos, T., Anagnostopoulos, C., and Hadjiefthymiades, S. (2011). An adaptive location prediction model based on fuzzy control. *Computer Communications*, 34(7):816–834.

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

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*.

Orlinski, M. and Filer, N. (2013). The rise and fall of spatio-temporal clusters in mobile ad hoc networks. *Ad Hoc Networks*.

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.

Perttunen, M., Riekki, J., and Lassila, O. (2009). Context Representation and Reasoning in Pervasive Computing: a Review. *International Journal of Multimedia and Ubiquitous Engineering*, 4(4):1–28.

Rabin, M. J. A., Hossain, M. S., Ahsan, M. S., Mollah, M. A. S., and Rahman, M. T. (2013). Sensitivity learning oriented nonmonotonic multi reservoir echo state network for short-term load forecasting. In *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 1–6. IEEE.

Rolim, C. O., Rossetto, A. G., Leithardt, V. R. Q., Borges, G. A., dos Santos, T. F. M., Souza, A. M., and Geyer, C. F. R. (2015). An Ubiquitous Service-Oriented Architecture for Urban Sensing. In *Agent Technology for Intelligent Mobile Services and Smart Societies*, chapter 1, pages 1–10. Springer Berlin Heidelberg.

Yu, P., Miao, L., and Jia, G. (2011). Clustered complex echo state networks for traffic forecasting with prior knowledge. In *2011 IEEE International Instrumentation and Measurement Technology Conference*, pages 1–5. IEEE.