Computer Networks 000 (2016) 1–16

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Situation awareness and computational intelligence in opportunistic networks to support the data transmission of urban sensing applications

Carlos Oberdan Rolim[a,*], Anubis G. Rossetto[a], Valderi R.Q. Leithardt[a], Guilherme A. Borges[a], Cláudio F.R. Geyer[a], Tatiana F.M. dos Santos[b], Adriano M. Souza[b]

[a] Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre – RS, Brazil
[b] Postgraduate Program in Production Engineering, Federal University of Santa Maria (UFSM), Santa Maria – RS, Brazil

ARTICLE INFO

Article history:
Received 15 January 2016
Revised 17 June 2016
Accepted 29 July 2016
Available online xxx

Keywords:
Smart cities
Urban sensing
Opportunistic networks
Situation awareness
Computational intelligence

ABSTRACT

Smart cities can be seen as large-scale Cyber-Physical Systems with sensors monitoring cyber and physical indicators and with actuators dynamically changing the complex urban environment in some way. In this context, urban sensing is a new paradigm that exploits human-carried or vehicle-mounted sensors to ubiquitously collect data to provide a holistic view of the city. A challenge in this scenario is the transmission of sensed data in situations where the networking infrastructure is intermittent or unavailable. This paper outlines our research into an engine that uses opportunistic networks to support the data transmission of urban sensing applications. It applies situation awareness and computational intelligence approaches to perform routing, adaptation, and decision-making procedures. We carried out simulations within a simulated environment that showed our engine had 12% less overhead than other compared approaches.

1. Introduction

According to [1], the physical world is becoming more and more saturated with the presence of a vast number of mobile devices. The integration of smart objects such as mobile and embedded computing devices with people and physical environments, which are typically tied by a communication infrastructure, inspired the development of the concept of cyber-physical systems (CPS). The concept of CPS encompasses systems ranging from a single smart house to an entire smart city. Smart cities are large-scale CPS with sensors monitoring cyber and physical indicators and with actuators dynamically changing the complex urban environment in some way [2]. Therefore, a key feature of smart cities is sensing of different aspects of the city in order to provide citizens with new services and improve their quality of life [3]. In order to facilitate this data collection, urban sensing applications are emerging as a promising way to “feel the pulse” of the city.

One challenge to implementing urban sensing applications is the transmission of sensed data using poor wireless infrastructures available in large-scale urban settings (i.e. with low coverage for the huge number of devices spread throughout the environment, low bandwidth available, wireless shadowing, frequent disconnections, etc.). Authors in [1,4,5] suggested that opportunistic networks could overcome a lack of connectivity in smart cities and, consequently, could be applied to support data transmission of CPS. In an opportunistic network, direct, physical contacts between nodes are opportunistically exploited to recognise and disseminate relevant information toward potentially interested nodes, without the need of centralised infrastructures or precomputed paths from source to destination [1].

In this paper, we propose an engine based on the opportunistic network paradigm to transmit data of urban sensing applications in scenarios where the networking infrastructure is intermittent or unavailable. The main differential of such engine is the application of situation awareness in conjunction with computational intelligence approaches to transmit data, perform routing, make adaptations and carry out decision-making. Furthermore, the engine will be used to underlie the opportunistic communication in Sensing module of our ubiquitous service-oriented architecture for urban sensing, UrboSenti [6].

* Corresponding author.
E-mail addresses: carlos.oberdan@inf.ufrgs.br (C.O. Rolim), agmrossetto@inf.ufrgs.br (A.G. Rossetto), valderi.quietinho@inf.ufrgs.br (V.R.Q. Leithardt), gaborges@inf.ufrgs.br (G.A. Borges), geyer@inf.ufrgs.br (C.F.R. Geyer), taty.nanda@gmail.com (T.F.M. dos Santos), amsouza@smail.ufsm.br (A.M. Souza).

http://dx.doi.org/10.1016/j.comnet.2016.07.014
1389-1286/© 2016 Elsevier B.V. All rights reserved.

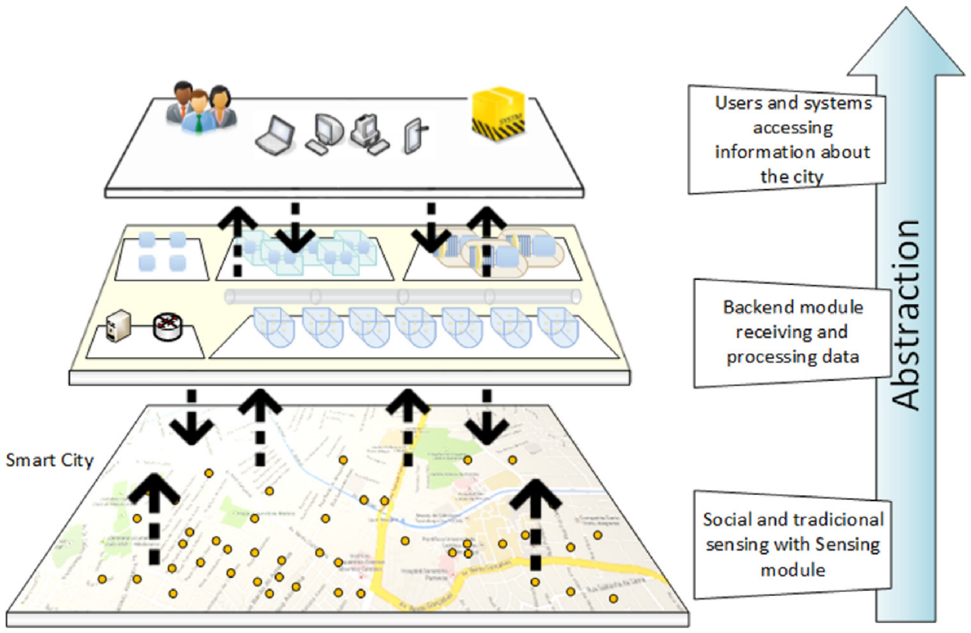

Users and systems accessing information about the city
Backend module receiving and processing data
Social and tradicional sensing with Sensing module
Abstraction
Smart City

Fig. 1. High-level view of UrboSenti.

It is our understanding that this is the first paper to adopt this kind of approach to smart cities with a focus on urban sensing applications. Additionally, the results of this study indicate areas in need of further research to 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 key issues about current computational developments. Section 3 provides a brief outline of some background concepts and related works. Section 4 describes the proposed architecture. Section 5 describes our experiments and analyses the results, and, finally, in Section 6 some conclusions are drawn and recommendations made for future research.

2. Problem scenario

Our research has been designed to address the problem-scenario of a smart city where several data sources are being used for sensing. Human-carried, fixed or vehicle-mounted sensors are utilized to obtain information about transit maps, air quality, noise levels, temperature, CO_2 concentration, etc. Moreover, data from social networks together with sensor data are crucial to understand the behavioural patterns of the city and to provide a holistic view of it.

Our Ubiquitous Service-Oriented Architecture for Urban Sensing (UrboSenti) is designed to collect, analyse, and provide feedback on sensed data obtained from several sources scattered around the city. The main purpose of UrboSenti is to provide support for the overall task of urban sensing. A high-level view of UrboSenti is depicted in Fig. 1.

Fig. 1 shows the division of UrboSenti into two key modules: the Backend module and Sensing module. Backend module operates in a data center infrastructure and, in short, is responsible for receiving sensed data, processing it and giving feedback to the public and other systems. Sensing module is responsible for urban sensing and encompasses activities involving intentional and non-intentional sensing. It operates in mobile devices (e.g. mobile phones, embedded systems in vehicles, etc.) and in fixed sensors scattered around the city. Vehicles and fixed sensors run a lightweight version of Sensing module that collects data without user interaction. In mobile phones, this module operates as an application that permits the users to report some events in the city or be configured to acquire data without user interaction when some threshold is triggered.

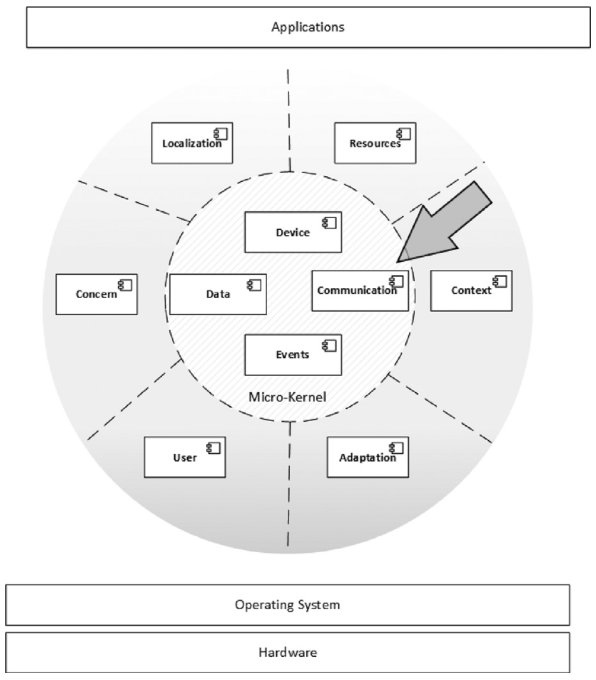


Fig. 2. Communication component highlighted in figure.

In both cases (with mobile and fixed devices), Sensing module has a microkernel with a set of components that can be plugged in "on demand" and are responsible for its essential features. These functions include Communication component (highlighted in Fig. 2) which is responsible for the management of all communication tasks.

The Communication component transmits sensed data using opportunistic networks paradigm when direct connection to the Internet is unavailable. In such approach, the contacts between nodes are used for data forwarding. The contacted node acts as a “data mule”, carrying the sensed data in local buffer until the next node is encountered. This process repeats until a node with an Internet connection is reached and data are sent to the Back-end module. Determining when to forward message to other node presents a challenge in this process. A wrong forwarding strategy could saturate the network, exhaust the nodes resources (e.g. power or buffer space) or even result in data loss.

To overcome this challenge, we are proposing an engine that applies *situation awareness*, a high abstraction view of context, in conjunction with *computational intelligence* approaches. We argue that an engine with these features could improve decisions about routing and data transmission of Communication component. In the next section, we describe the concepts used in this engine.

3. Background and related works

According to Celino and Kotoulas [7], urbanization has dramatically increased over the past few years, and forecasts show that migration to urban areas will continue to increase. The population concentration within cities poses numerous challenges in terms of both city governance and people’s lives. As a consequence, smarter solutions are necessary to better address emerging requirements in urban environments. The concept of “smart cities” is a response to these challenges [8]. Smart cities are typical examples of distributed CPS since they integrate different domains in order to create sophisticated solutions and applications that may or may not interact with the city’s residents [9]. In such scenarios, the exploration of users mobile devices and fixed sensors along to the city to collect data from physical world to be processed, stored and exploited in cyber world is an emerging trend called urban sensing.

The literature depicts a number of initiatives in urban sensing. Some examples include MetroSense [10], AnonySense [11,12], Medusa [13,14], PRISM [15], MobiSens [16], Pogo [17], CenceMe [18], SmarterSantander [19] and UrboSenti [6]. Each initiative proposes a different approach to collect data from city, analyse it, and give feedback to citizens. In this paper, our focus is not on such urban sensing solutions, but in the engine that runs in the sensing module of UrboSenti. As presented above, this engine applies situation awareness and computational intelligence approaches in Opportunistic Networks to improve the routing of sensed data. These concepts are further explained in the next sections.

3.1. Opportunistic networks

Opportunistic networking is a technology with good prospects to realize the ubiquitous vision [20]. This new emerging paradigm is sometimes referred in the literature as a subclass of Delay-Tolerant Networks (DTN). Opportunistic Networks seeks to simplify the complexity at the network layer by removing the assumption of physical end-to-end connectivity while providing connectivity opportunities for pervasive devices when no direct access to the Internet is available. It represents the first attempt to close the gap between human and network behaviour by adopting a user-centric approach to networking and exploiting node mobility for users so that it can be regarded as an opportunity – rather than a challenge – to improve data forwarding [21].

In opportunistic networks, the nodes are assumed to be mobile, and the forwarding of data is based on the Store-Carry and Forward (SCF) concept [22] where nodes act as “data mules” and data forwarding occurs at each nodes’ contacts. One fundamental problem in SCF is how to select effective data mules to carry data from their source to their destination in a suitable way (i.e. with high delivery rate, low latency and low overhead), since end-to-end paths might be absent for the lifetime of the message [23]. Several measures have been employed to overcome this problem. These measures involve adopting different approaches, ranging from “naïve” (i.e. using no or slight contextual information) to “intelligent” (i.e. using contextual information such as location, history, connectivity or social information for intelligent decision-making). As example of initiatives that uses these approaches we could cite: Epidemic [24], Spray&Wait [25] (and the Spray variants like Spray&Focus [26], Fuzzy-Spray [27] and others), Prophet [28], BubbleRap [29], dLife [30], SPRINT [31], AFRON [32], DRAFT [33], Cartoon [34], CAR [35], HiBOp [36], CiPRO [23], RAPID [37], Propicman [38], dLife [30], and SCORP [39]. Due to space limitation and this paper’s scope, we will not present more information about these initiatives – for further information see [40] and [4].

These existing initiatives in opportunistic networks do not focus on smart city applications. Conti et al. [41] indicated that smart cities are heterogeneous and dense scenarios, with constant topological changes with no pre-defined pattern due to the mobility of nodes. Consequently, data communication occurs spontaneously, resulting in a huge amount of small streams of data. Therefore, such initiatives cannot be used “as-is”, without adaptation, as underlying paradigm for data communication in an urban sensing scenario.

To overcome such limitations, we are proposing our engine. It uses some concepts of context-based information for routing decisions like Cartoon [34], but rather than just rely on instantaneous information, it makes decisions based on predictions. Moreover, our work is sited in the same area of computational intelligence as the CAR protocol [42]. However instead of depending on Kalman Filters for prediction and a multi-criteria decision theory when choosing the ideal next hop for the message, in our work we have employed fuzzy logic for decision-making and neural network for prediction.

3.2. Situation awareness

Situation awareness (SAW) is a computing paradigm which uses context-based data to sense and understand the current situation and to forecast future requirements. According to Ye et al. [43], a situation is an abstraction of events occurring in the real world derived from the context and hypotheses based on the relationship of observed context to factors of interest to designers and applications.

The first formalization of SAW was a 3-level model proposed by Endsley [44]. The levels of the model were perception, Level 1 SA which involves the monitoring and detection of states of elements; comprehension, Level 2 SA which concerns the formation of a comprehensive picture of the elements by integrating different kinds of context-based information; and projection, Level 3 SA which is the highest level of SAW and involves the projection of the future actions of the elements. According to the author, situation awareness provides a more holistic view that improves contextual representation and reasoning.

In our work, the proposed engine collects internal and external context data of node and environment to derive a high abstraction view or situation. It applies the 3-level model of Endsley [44] to support decision-making processes. The main advantage of this approach is the human-friendly representation of context data and abstraction of complexity to read and to infer about those data [43].

As examples of works in this area, we could cite the RCSM Middleware [45] and European Project CASCADAS [46]. We could not find any initiatives of SAW in opportunistic networks in urban sensing area.

3.3. Computational intelligence

Although used in different contexts, no widely accepted definition of the term computational intelligence (CI) exists [47]. James

Bezdek popularized the term who defined it as a discipline of artificial intelligence composed of computer systems that use numeric data, recognise patterns, display computational adaptability and fault tolerance, and commit errors at a rate approximating human performance [48]. As IEEE Computational Intelligence Society[1] indicated, the concept of CI involves different topics of artificial intelligence which is composed of the subareas of neural networks, fuzzy systems, genetic algorithms and evolutionary computing including swarm intelligence. In this context, the component that is being proposed will explore two approaches in particular: neural networks for predictions of future situations and fuzzy systems for the treatment of uncertainty and decision making. Genetic algorithms and evolutionary computation was not be used because they are more suitable for optimization problems and classification [49].

3.3.1. *Artificial neural networks*

Artificial Neural Network (ANN) are mathematical models that simulates a highly inter connected, parallel computational structure of the brain. It is considered a sub-area of computational intelligence. Henceforth, in this text the terms artificial neural network and neural network are used interchangeably.

Basically, there are two types of neural networks: feedforward and recurrent. In Feedforward Neural Networks (FNN), an activation signal is propagated through the neural network from input units to output units. In Recurrent Neural Networks (RNN), the values of output units are used as new input creating a directed cycle. With this arrangement, the network creates an internal memory state which allows the network the ability to handle problems by predicting tasks for the next state based on past states [50]. For predicting tasks, RNNs, in general, achieve better results in prediction of time series than feedforward networks [50,51].

Despite this widely acknowledged potential, RNNs are difficult to train by gradient-descent-based methods, which attempt to iteratively reduce the training error. To overcome such limitation a new approach called Reservoir Computing (RC) was proposed. The main differential of the RC paradigm is related to setting up RNNs in the following way: (i) an RNN is randomly created and remains unchanged during training. This RNN is called the reservoir. It is passively excited by the input signal and maintains in its state a nonlinear transformation of the input history; (ii) the desired output signal is generated as a linear combination of the neurons signals from the input-excited reservoir. This linear combination is obtained by linear regression, using the teacher signal as a target [52].

RC methods have quickly become popular and today two kinds are prominent: Echo State Networks (ESN) proposed by Jaeger [53] and Liquid State Machines (LSM) proposed by Maass et al. [54]. Both methods are based on the usage of a reservoir of neurons, but LSM uses a more sophisticated synaptic models which are usually more difficult to implement to correctly set up and tune and more expensive to emulate on digital computers [52]. In contrast, ESN uses a unique dynamic reserve pool structure that allows the network to have effective short term memory capacity and to show better performance in forecasting with low computational costs [55].

ESN is a three-layered recurrent network with sparse, random, and, most importantly, untrained connections within the recurrent hidden layer. As a 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 randomly generated rather than trained. This approach significantly reduces the learning process when compared with other algorithms (e.g. back propagation through time) and, therefore, requires low computational costs to implement. This factor led us to use ESN as an essential technique for forecasting of our engine.

ESN has been employed to solve practical problems in various domains. ESN has been used to forecast predictions for power plants and power grids [56–58], flue gas turbine condition trends [55,59], wind power generation [60], exchange rate forecasting [61] and for credit rating systems citeBozsik2012. ESN was also applied to predict security events in computer networks, and more related to our work, Yu et al. [62] use ESN to predict mobile communication traffic in a cellular network. However, at present, we cannot find any work that applies ESN in urban sensing applications.

3.3.2. *Fuzzy logic*

Fuzzy logic is another sub-area of computational intelligence used by our engine. The fuzzy set theory was proposed by Zadeh in 1965 as an extension of multi-valued logic. It has been described as a precise logic of imprecision and approximate reasoning. A fuzzy set A in X is characterized by a membership function fA(x) which associates with each point in X a real number in the interval [0,1], with the values of fA(x) with x representing the “grade of membership” of x in A. Thus, the nearer the value of fA(x) is to unity, the higher the grade of membership of x in A citeZadeh1965. The shape of the membership function defines the fuzzy set and is dependent on the purpose of set. The most commonly used membership functions are triangular, trapezoidal, generalised bell, sigmoidal and gaussian.

The Fuzzy Inference System (FIS) is a popular computing framework based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning [63]. An FIS is a system that uses fuzzy set theory to map inputs to outputs. The mapping then provides a basis from which decisions can be made. The process of fuzzy inference involves the determination of a set of ‘if-then’ fuzzy rules, the fuzzification of inputs using membership functions, the application of rules and fuzzy operations to generate consequents and to get an output distributions and finally the defuzzification of output distribution in case of crisp output.

Two types of inference methods are used in FIS: Mamdani and Takagi-SugenoKang type (or simply Sugeno). The Mamdani method of inference expects the output membership functions to be fuzzy sets. After the aggregation process, a fuzzy set is established for each output variable that needs defuzzification. Sugeno FIS is similar to the Mamdani method in many respects. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator are exactly the same in both methods. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant.

We are using fuzzy logic because it is able to support real-time decisions about the context data of nodes once such data has some degree of uncertainty and vagueness. For this reason, conventional logic may lead to completely wrong decisions owing to uncertainty within context data. Fuzzy logic is a viable alternative to reasoning and making rational decisions with imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and degrees of probability [64].

Fuzzy logic has been successfully applied in a wide variety of fields such as automatic control, data classification, expert systems, time series prediction, robotics, and pattern recognition and decision making [65]. In ubiquitous computing, it has been used for different purposes such as location prediction [66] and context inference [67]. In the field of opportunistic networks, the closest initiatives are AFRON [32], Fuzzy-Spray [27], FuzzyCom [68] and

[1] http://cis.ieee.org/.

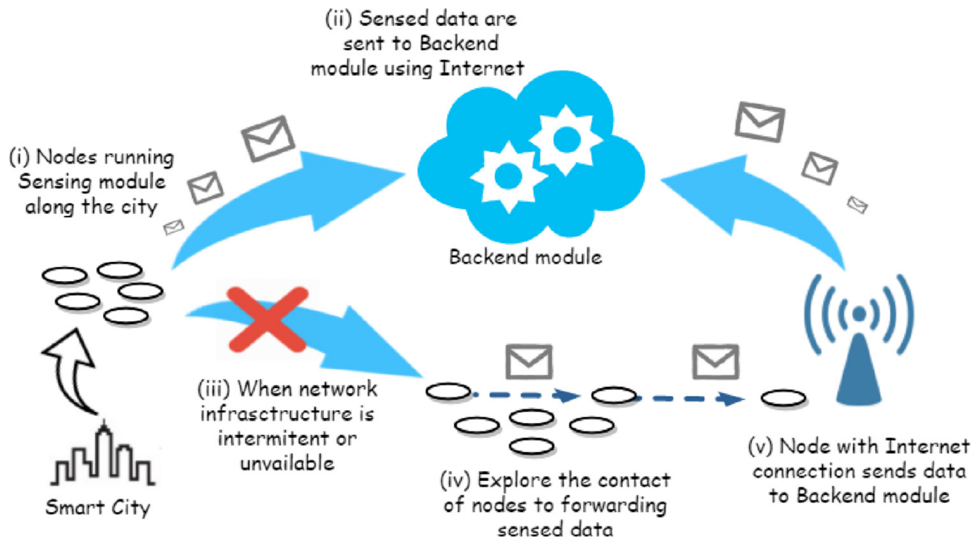

(ii) Sensed data are sent to Backend module using Internet
(i) Nodes running Sensing module along the city
Backend module
(iii) When network infrasctructure is intermitent or unvailable
Smart City
(iv) Explore the contact of nodes to forwarding sensed data
(v) Node with Internet connection sends data to Backend module

Fig. 3. Engine behaviour.

Adaptive Fuzzy Spray&Wait [69]. However, none of them was built with focus in urban sensing area.

4. Proposed solution

As shown in Section 2, in this work we are describing our engine used to underlie communication data when opportunistic networks paradigm is used by Sensing module of UrboSenti. In this section, we describe how the engine works, its conceptual model and then outline its inner architectural features.

4.1. Engine dynamics

As we explained above, UrboSenti is an architecture that encompasses the overall urban sensing tasks. It is composed of two main modules: Backend and Sensing.

Sensing module runs in each sensor node in a smart city (Fig. 3). These nodes collect data about city intentionally or unintentionally, without user interaction (i). In normal operation mode, Sensing module collects data and sends it to be processed by Backend module using its internal communication component (ii). Communication component provides methods for transmission of sensed data by means of the available network infrastructure, such as IEEE 802.11b/g/n (both structured and ad-hoc), GPRS/EDGE/3G and Ethernet using traditional TCP/IP stack.

In some parts of the city, the network infrastructure could be overloaded, intermittent or unavailable. In this case, communication component of the sensing module receives an alert of other internal components indicating that it needs to adapt its behaviour to send data without network infrastructure (iii). In this case, it starts to use the opportunistic network as paradigm for data transmission. In this operation mode, instead of sending data directly to the Internet, the node stores it in a local buffer. When another node is encountered, the communication component employs our engine to decide if the stored data should be forwarded or stay in local buffer until next encounter (iv). This process repeats until a node with an Internet connection is reached and sensed data is sent to be processed by Backend module (v).

To support such decision-making process about forwarding, the engine uses some conceptual models detailed in next section.

4.2. Conceptual models

As previously presented, a challenge in opportunistic networks concerns the right time to forward message to contacted node. To facilitate such decision-making process, our proposed engine applies concepts of situation awareness and computational intelligence. To support such tasks, we developed some conceptual models that will be implemented later.

The first model is Situation awareness (Fig. 4). The main function of this model is to understand what happens when the node and project adapt proactively. It is based on a 3-tier Endsley's situation model presented in Section 3.2. This model groups internal context data such as power, buffer usage, traveled distance from nodes and external context data from scenario such as number of neighborhood nodes, type of contacted node, etc. in a set called low level context. Such data provide the basic perception of situation (first level of Endsley's situation model) and are used as input for the situation comprehension (second level of situation). To this task, a repository of rules is used to characterize the current node situation. Once the current situation is acknowledged, it is stored to be used in the future as a reference of past situations and also is used as a basis to predict future node situations (third level of situation). To define future situations, a set of rules and a computational intelligence approach with support for prediction is applied (for such tasks, the prediction model described next will be used).

At this point, the engine becomes aware of the past, current and future node situations and such information is used to decide about routing. Since all information has some degree of uncertainty and vagueness, fuzzy logic is applied to facilitate the decision-making (the decision-making model is presented next).

As a result of the overall processes, an indication about routing decision is produced.

The Prediction model (Fig. 5) is used by situation awareness model support prediction of future situations. A recurrent Neural Network (NN) is used for this purpose. We have chosen NN because of its capacity to solve non-linear problems; it has universal function approximators suitable for prediction. Each node is responsible for training and running its own NN. This approach allows us to ensure that each node has a suitable NN to suit its needs. The current and past low-level context-based data are used as input for NN. The NN starts the training phase by testing several configurations from a configuration repository and seeking to

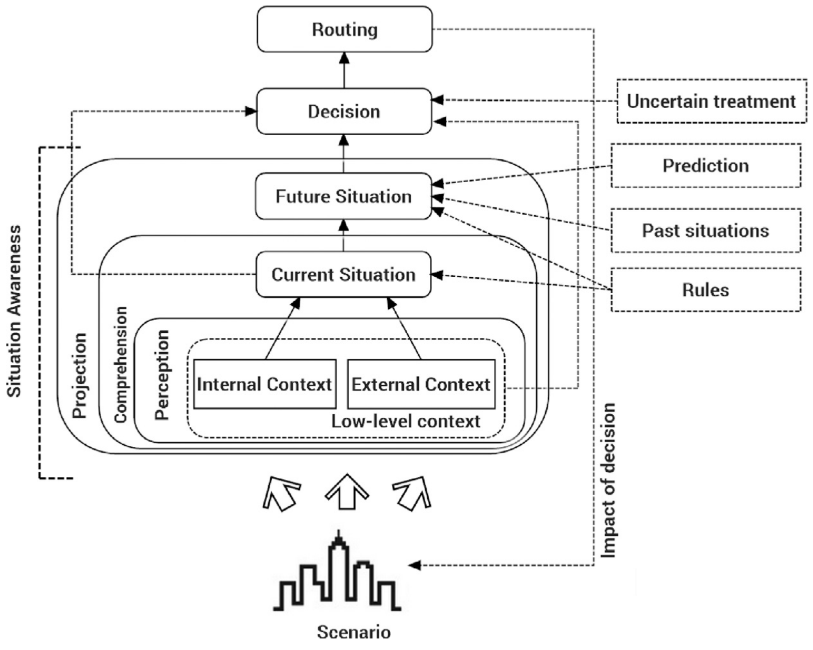


Fig. 4. Situation model.

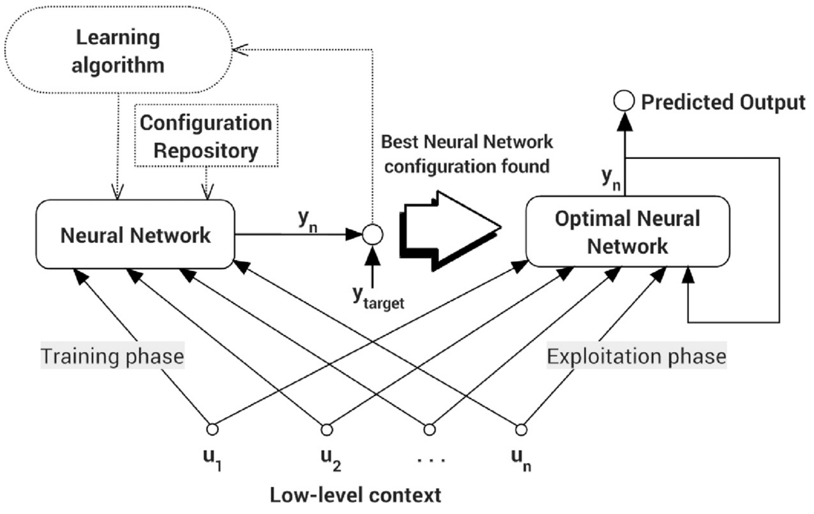


Fig. 5. Prediction model.

find the optimal Network (with the lowest Root Mean Squared Error – RMSE). When an optimal network is found, it is used for prediction in the next phase, exploitation. In the exploitation phase, the current low-level context data is pumped to the optimal NN found in the previous phase. The outputs of this process are newly predicted, low-level context data that probably characterize the future situation of a node. It is an essential requirement for NN to ensure that computational costs are kept low due to the power and processing constraints of the mobile nodes.

The Decision-making model (Fig. 6) is designed to define the potential of a node to be an effective data mule. As it handles uncertain data, fuzzy concepts are applied. Precise and imprecise low-level context-based data (from current and projected future situations) are used as input. These context data are fuzzified using membership functions and the inference runs with application of 'if-then' fuzzy rules. The output of inference is defuzzified and the potential of node to be a good data-mule is generated.

4.3. Engine architecture

Our engine makes use of dynamism and the models depicted above. Its internal architectural features are outlined in Fig. 7 and its behaviour is explained below.

The engine starts with the *Contextual Information* that represents information about the context of node. At constant time intervals, *Context Collector* collects the data and stores them in *Contextual Graph*, thus creating a new Layer 2 vertex. The *Contextual Graph* component manages all the data storage. The Neo4j[2] graph database was used as the underlying software for this purpose. Its main function is to store instantaneous and predicted context

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

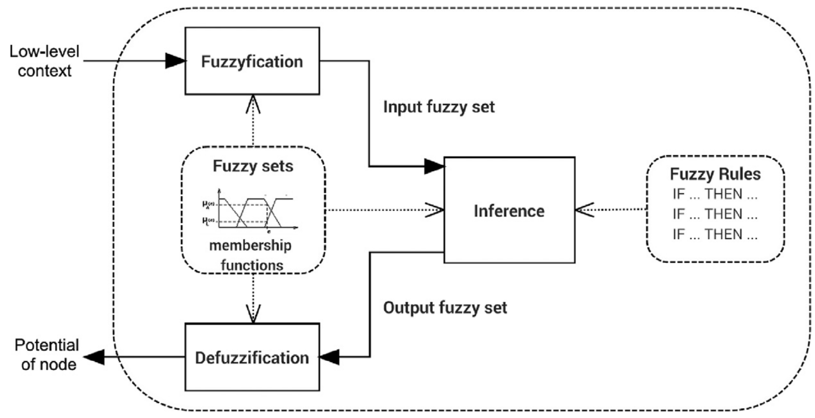


Fig. 6. Decision-making model.

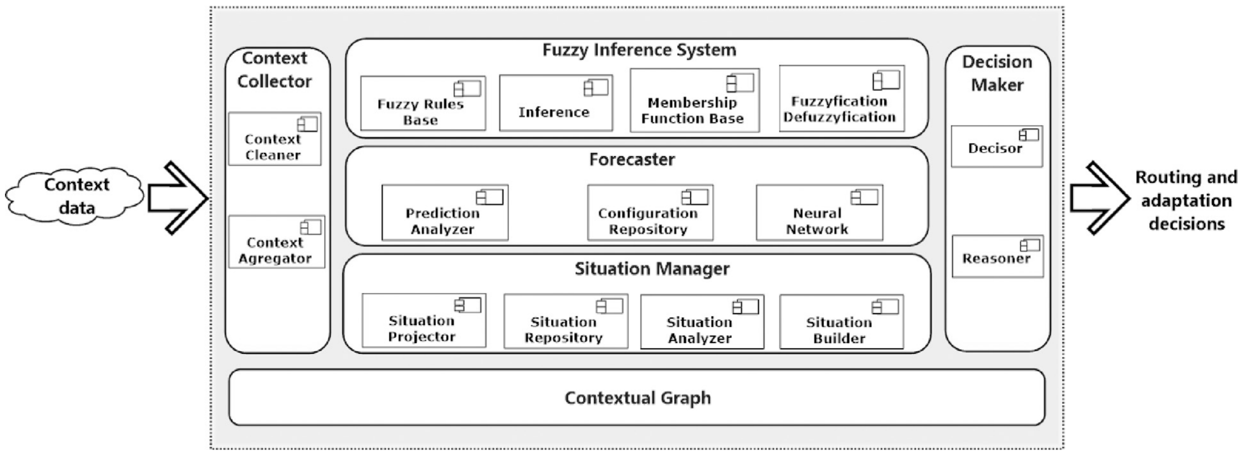


Fig. 7. Engine architecture.

information. In Contextual Graph, the vertices of the graph are structured in layers: Layer 1 stores basic information about the node (i.e. node name, address, network interfaces, etc.), 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, etc.) that will be used as historical values to prime the Forecaster component, and Layer 3 stores predicted context values that are periodically generated by Forecaster. Moreover, we used edges to represent the contacts between the nodes. Fig. 8 shows an example of partially stored data in one node of the network.

The *Situation Manager component* implements our situation awareness model. It draws on data from the Contextual Graph to build, analyse, project and create a repository of situations. The information generated by this module is used later by the Decision Maker. It also runs maintenance routines like pruning old data and invoking Forecaster for prediction. At fixed time intervals, the Situation Manager retrieves context data from the Contextual Graph and with the aid of a set of rules stored in its internal situation repository, it seeks to determine (“build”) the situation of the current node. The identified situation (e.g. the node is sensing low battery power and high buffer usage) is analysed, and if it finds that some action needs to be taken, this is reported to the Decision Maker. If a situation cannot be identified, an unknown situation is found. Thus, a new set of rules that characterise this situation is created “on the fly” and stored in the repository for future use. When the Situation Manager component detects a sufficient amount of context information, it is able to project a future situation. The Forecaster component is invoked for this purpose.

The *Forecaster component* implements our prediction model, so that it can predict the probable values that will characterize a future situation. When carrying out this task, it uses context values that are stored such as the values in Layer 2 in the Contextual Graph which provides historical data to train ESN (i.e. node power, current position, etc.). The low computational costs involved in training the neural network allows us to enable each node of the network to build its own ESN (Neural Network) with the most appropriate configuration for its context. This is accomplished by testing different internal parameters of ESN (i.e. the size of reservoir, sparsity of the reservoir, spectral radius and leak rate) with different values until the one with the lowest MSE is found. When the optimal neural network is found, its configuration is stored. At this stage, the optimal network is ready to predict future values in the exploitation phase. During the exploitation phase, the structure with historical data is used to “pump” the best network which was been found and saved in the previous phase with historical context data to activate the internal reservoir. Some stages later, the input data are switched off to allow the network to predict values in a self-recurring way. The predicted values are stored in the Contextual Graph as Layer 3 vertices. At this stage, the component sets an internal variable to indicate that this node is now running in smart mode. In smart mode, all the decisions of the Decision Maker are made on the basis of past, current and predicted data to improve the accuracy of routing and adaptation decisions. In “dummy” mode, only the current situation is used.

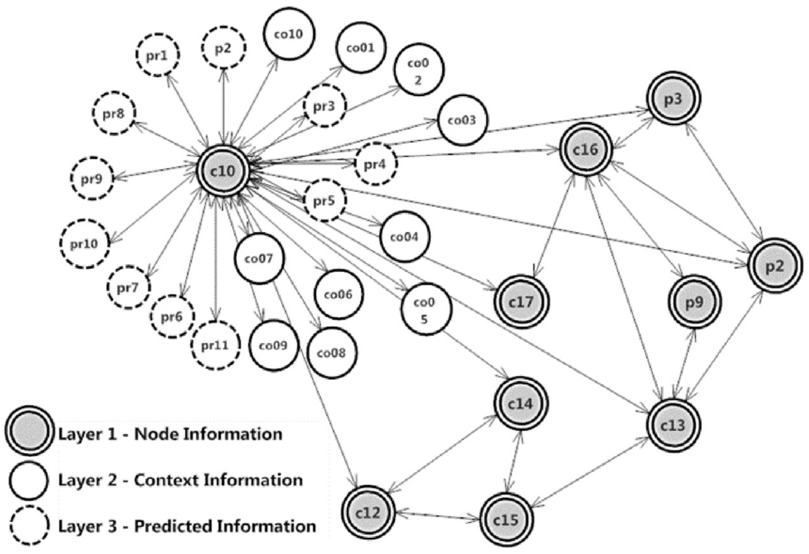


Fig. 8. Example of conceptual graph structure. In detail stored data in node c10.

The *Decision Maker*implements our Decision-making model. It runs at constant time intervals to decide if some internal parameters need to be adjusted (such as the buffer scheduling policy, maximum size of messages, time to live of new messages, etc.) or if a "trap" should be triggered to attract the attention of an external component of the micro-kernel of the Sensing module (e.g. to change the configuration of the network interface, to perform some adaptation action, etc.). Decision Maker is also invoked when the current node contacts 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 an effective "data mule". We employ the term potential to represent the capacity of the node to carry and deliver message to its destination of forward it to a closest neighbour. The strategy used in Decision Maker 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). FIS is used to calculate the potential of each node. All context values (current, past and predicted) of the current and contacted node from Contextual Graph are used as input for the FIS. It uses its internal components and fuzzy rules to calculate the potential of each node.

5. Simulation and experimental results

5.1. Implementation of the main components

The implementation of main components of the engine is described below.

5.1.1. Forecaster

The ESN used in Forecaster was designed by using ESNJava[3]. It provides a graphical interface making ESN networks easier to handle and API easier to embed ESN in Java applications. However, the ESNJava only handles situations where a sequence of values that must be learned (i.e. teacher enforced) are received as input and the network is trained to reproduce the desired dynamic properties for this original sequence. In other words, the ESNJava only seeks to reproduce learned input and does not provide predictive support. This restriction was overcome by making changes in the source code of ESNJava. The original code of ESNJava was changed to introduce support for the network itself so that it could predict the output used as the next input stage in a self-recurring way (as introduced in the description of the prediction model in Section 4).

As each node executes its own neural network and results in a specific configuration, we have not shown the configuration results for each node, but only the results from the average of all the nodes. On average, around 104 epochs were used in the training phase and 54 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 neurons 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 leak rate = 0. Fig. 9 graphically represents the process of selecting the best network configuration of one node. For this node, the best network configuration had a network size of 10 and spectral radius of 0.77.

5.1.2. Decision maker

The FIS used in Decision Maker was implemented using JFuzzyLogic library[4]. The following linguistic variables were defined to form the FIS that was used to calculate the node potential: 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. These variables represent different aspects of the context and each has linguist values ("low", "medium", "high") associated with a Gaussian membership function with center and width values scaling in accordance with the magnitude of the context data. The variable potential which is used as the output of FIS employs a three triangular membership function with values ranging from 0 to 100. The COG (Center Of Gravity) was used as a defuzzification method and the default value is 0 when no rule is activated in the defuzzification.

The fuzzy inference rules were defined in compliance with Fuzzy Ccontrol Llanguage (FCL) and adopted the form IF variable IS property THEN action. An example of a rule that was used in our FIS, is IF power IS low OR power IS medium THEN potential IS low. Fig. 10 shows some of the used membership functions and the

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

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

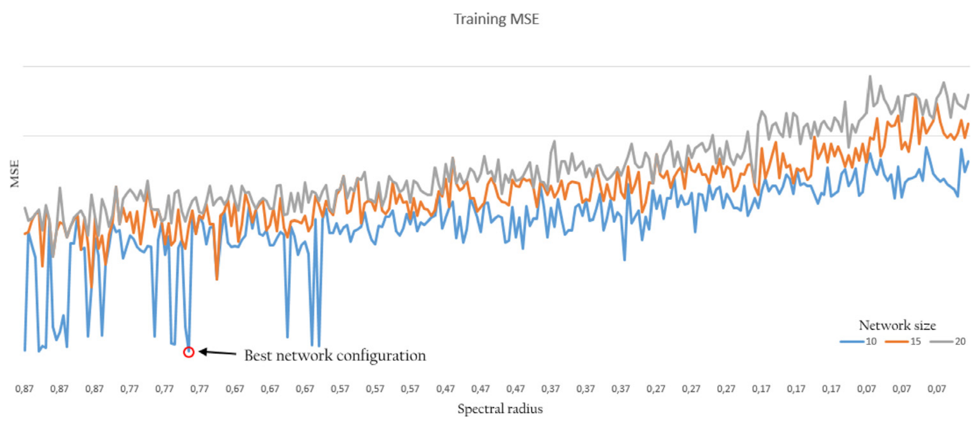


Fig. 9. Selection of best network configuration.

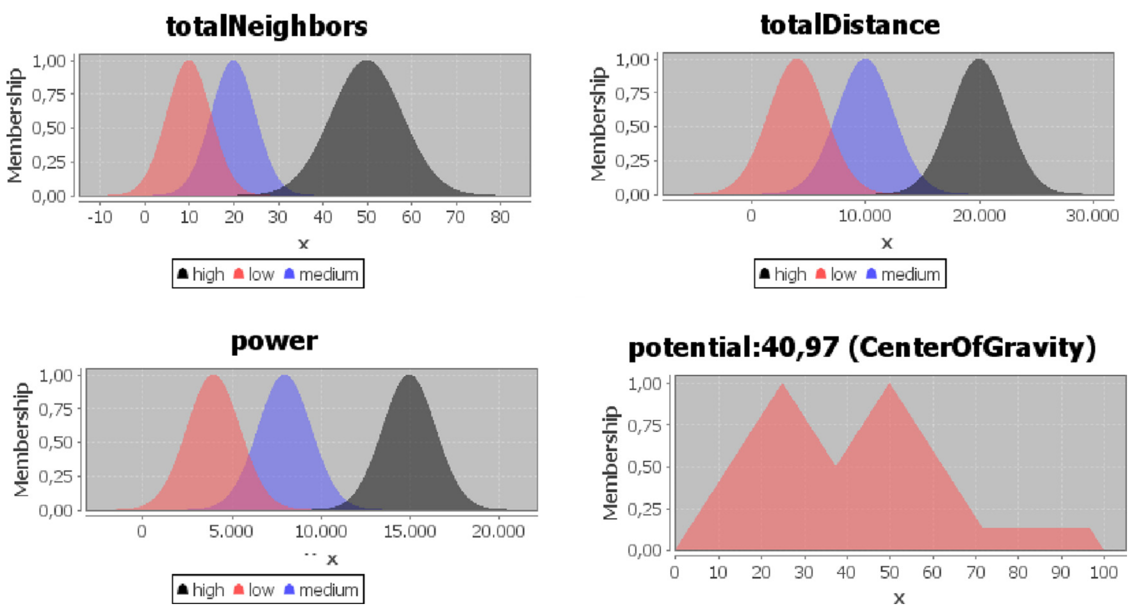


Fig. 10. Example of membership functions used in decision maker module.

defuzzified potential of one node indicating that it has a potential of 40.97.

5.1.3. Situation manager

Java Drools[5] was used for the Situation Manager and this employs an inference rule system through an enhanced implementation of the Rete algorithm[6].

Since there are a large number of similar rules with different values, we decided to make use of the Drool Decision Table. Decision tables are a precise yet compact way to model complicated logic. It works like 'if-then-else' and switch-case statements and associates conditions with actions to perform. However, unlike the control structures found in traditional programming languages, decision tables can associate many independent conditions with several actions without several 'if-then' conditions. A decision table consists of three parts: (i) condition rows that list conditions relevant to decision; (ii) action rows that identify actions that result from a given set of conditions; and (iii) rules which specify which actions are to be followed for a given set of conditions.

[5] http://www.drools.org/.

[6] A Rete algorithm is a pattern matching algorithm for implementing a production rule system.

As the Situation Manager is still under construction, the following experiments were only carried out with a basic situation set (as described above), without incorporating new situations into the repository.

5.2. Simulation setup

The main modules were implemented to determine the functionality and performance of the proposed engine and some simulations were conducted with the aid of the Opportunistic Network Environment (ONE) Simulator.

As illustrated by [70], a typical large-scale urban sensing scenario is formed by fixed nodes installed at strategic locations around the city and mobile nodes carried by humans (e.g. smartphones) or by vehicles (e.g. cars, buses and trains). Thus, to simulate data transmission in the city, two groups of nodes were created: mobile and fixed. The mobile group consists of pedestrians and vehicles. The fixed group consists of static sensors and access points strategically placed in the city.

The simulation runs a sensing application in mobile nodes and static sensors along a city with 490 km^2 of area. When an event is sensed, the collected data are forwarded between nodes using the opportunistic network paradigm until an access point is reach. The access point acts as a "bridge" between the city and the Inter-

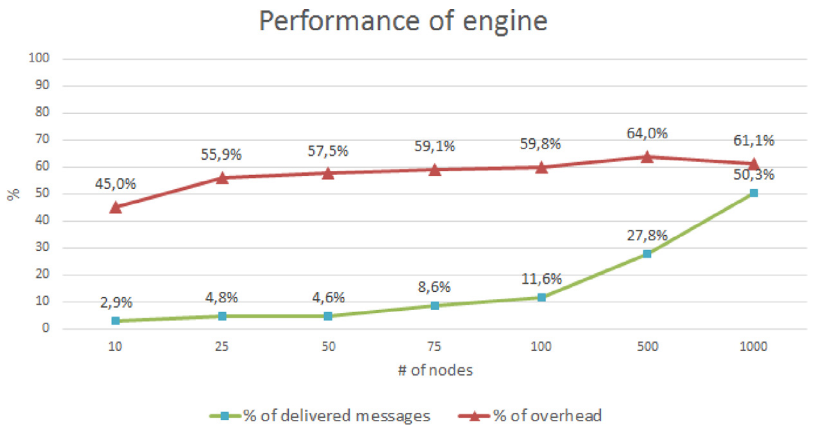


Fig. 11. Performance in scenarios with different numbers of nodes.

net, which sends all received data to be processed by the Backend Module hosted in a data center infrastructure.

In the simulation, pedestrians and vehicles moved using the *Shortest Path Map Based Movement model* (this model uses Dijkstra's algorithm to find shortest paths between current location to a randomly selected destination, by using the roads or paths). This model was also used in works of [30] and [71]. The pedestrian nodes moved between 0.5 and 1.5 km/h, and had a Bluetooth device with a radio range of 5 m and transmission speed of 2 Mbps and a Wi-Fi interface with a range of 20 m and transmission speed of 10 Mbps. The vehicle nodes moved between 10 and 50 km/h and had a Wi-Fi interface with same transmission and range of pedestrian nodes. Both pedestrian and vehicles had a buffer size of 50 M. A growing number of mobile nodes were used for all the simulations: 10 nodes of each group in the first test case, 25 for the second, and 50, 75, 100, 500 and 1000 for each consecutive scenario.

In the fixed group, we placed 50 static sensors around the city. Each sensor had one Wi-Fi interface with a range of 30 m and transmission speed of 10 Mbps. Ten access points were used. Each one had Wi-Fi with the same configuration of the static sensors and a wired interface of 1 Gbps that simulated the connection to the Internet. The data buffer of static sensors was set with 256 M, and the buffer of access points was set with 512 M.

A total time of six hours (21,600 s) was adopted for all the simulations. On average, the nodes generated about one message of sensing data every 25 to 35 s. A total of 711 messages was generated in each simulation. Message sizes were set at a uniform distribution between 100 KB and 2 MB. The message lifetime (TTL) was set to 24 min (1440 s) to prevent nodes from carry old messages for a long time.

The same power restrictions that were used by Rodrigues-silva et al. [71] was applied in the simulation. All mobile nodes started with a fully charged battery of 19,080 Joules which corresponds to the power of typical cellular battery of 5.3 W/h and 3.7 V and a recharge was set randomly in intervals between 4 and 4.5 h. The energy spent on scanning the other nodes was 0.092 mW/s, and the energy required for sending and receiving messages was 0.08 mW/s. Only fixed nodes were set with unlimited energy, without need of recharges, since they are permanently connected to the power grid.

5.3. Evaluation metrics

To evaluate our simulations, we adopted the same performance metrics used by Karamshuk et al. [72] and supported by ONE Simulator: Started, the number of transmissions started between network nodes; Created, the number of messages created during simulation excluding replicated messages; Relayed, the number of successful transmissions between nodes; Delivered, the number of successfully delivered messages; and Overhead ratio, the estimated number of extra messages needed by the routing protocol for actual delivery of the data. It is defined as the following formula: (Number of messages relayed - Number of messages delivered) /(Number of messages delivered).

5.4. Experimental results

5.4.1. Experiment 1 - performance evaluation

The purpose of the first experiment was to assess the performance of the proposed engine to handle the total of 711 messages generated and the impact of the number of nodes on the percentage of delivered messages and overhead. As explained above, we tested different scenarios ranging from 10 to 1000 nodes. The results are shown in Fig. 11.

Fig. 11 shows on average an incremental rise in the percentage of delivered messages related to the increase of nodes in scenario. When we increased the total number of nodes from 10 to 1000 (an increment of 1000% in the number of nodes), the percentage of delivered messages increased from 2.9% to 50.3%, an increase of 1634%. If we compare the scenario of 50 to 100 nodes (an increment of 100% in the number of nodes), the percentage of delivered messages increased from 4.6% to 11.6%. It corresponds to an increment of 152%. But, if the number of nodes were doubled again, from 500 to 1000, were an increment of just 80% in number of delivered messages (from 27.8% to 50.3%) In other words, the incremental rise in the percentage of delivered messages not necessary follows the perceptual increment of nodes.

The success in the routing decision strategy is strengthened by the overhead based metric. The scenario with 1000 nodes had an increase of 80% in the number of delivered messages to the scenario with 500 nodes, but with 4% less overhead. That is, even when the number of nodes were increased by more than 100%, the overhead remained low. This indicates that as the number of nodes increase, the engine improves the decision-making process due to the increase in contacts which increases the amount of available context data used to characterize the situation and the model's ability to forecast. Thus, a low overhead indicates that effective data mules were selected.

Additionally, when the number of nodes is low (in the case of scenarios with 10 and 25 nodes), the overhead percentage is significantly higher in comparison to scenarios with 1000 nodes. With a low number of nodes and fewer contacts, three issues occur: (i) messages stay in local buffer longer, resulting in delay of delivery or discard of the message due to expiration of TTL; (ii) the relay of messages is high and sometimes the same message returns to an already visited node; and (iii) the decision making process of the engine makes flawed forwarding decisions due to limited external context data (exchanged between nodes contacts). In future versions of the engine, we will try to overcome such issues using a variable TTL of message according to some context data such as detected number of contacts in last x minutes or sparsity of contacts and the implementation of a historical of visited nodes embedded in the message header to avoid messages revisiting nodes. In this case, we will need to verify the impact of such an approach. However, it is believed that these rates of delivered messages and overhead supposedly could be further improved through the conclusion of Situation Manager and inclusion of rules that allow a dynamic adjustment of the internal parameters of the component, such as buffer and TTL management.t.

5.4.2. *Experiment 2 - comparison of ESN with other neural networks*

In this experiment, we attempted to determine the effects of the Forecaster equipped with ESN in comparison to other low computation cost approaches. We selected two kinds of NN which have a reliable predictive power and were previously tested in our previous paper [73]: SVM (Support Vector Machines) and NARX (Nonlinear Autoregressive model with eXogenous input).

When conducting this experiment, each kind of such NN was “plugged” into the Forecaster and the simulations run. With ESN, each node trained and formed its own instance of ESN Neural Network. It was possible to do due to low computational cost of this approach. In the case of SVM and NARX, the same situation is unsuitable. Despite the findings in the literature that both are computationally effective, in our experiments SVM and NARX consumed a lot of memory and processing power. To solve this problem, we ran a fixed configuration for all nodes.

The ESN was used as discussed in Section 4. In the case of the SVM, we used Weka[7] which implements the SMOreg algorithm. Weka is a collection of machine learning algorithms employed for data mining tasks. We used the complexity constant C = 1.0 with standard normalisation. In the case of the RegSMOImproved optimizer, we used the Epsilon parameter in Epsilon-insensitive loss function = 0.001 and the Kernel with default values. The number of units was set to forecast 10 steps ahead. The best result in one node of network occurred with a RMSE (Root Mean Squared Error) = 5.48 at 10th step ahead.

NARX was implemented by Matlab R2013b[8] with 10 hidden layers and 2 delays. We trained NARX with the same input and desired output as were used in SVN. The forecast was set at 10 steps ahead. As NARX is a recurrent neural network, we used the last output to provide the network with feedback. With this configuration, the best result was an MSE of $9.0167e^{-06}$ with 100 epochs in the “validation” phase. All the values of the residual autocorrelation were in the confidence interval, which suggests that past errors do not distort the current prediction. After having trained and validated the network, we integrated the Matlab script with the simulator that was used by means of Matlabcontrol API[9].

Fig. 12 shows the results and the comparison of the performance of the proposed engine when each approach was adopted in different scenarios.

On the basis of the results, we formed the graph shown in Fig. 12 and examined the performance of the proposed engine when each approach was adopted in different scenarios.

In the scenario with 1000 nodes, the engine which ran the Forecaster equipped with ESN had 62.5% more delivered messages than SVN and 49.2% more than NARX. Compared with NARX in the same scenario, ESN had 1800% less overhead. The low overhead of ESN with respect to SVN was also observed in a scenario with 1000 nodes with 1222% less overhead. As can be observed, ESN was the best approach, SVN second and NARX was the worst.

One factor not shown in the chart is the computational cost of ESN. Even when several different configurations to find the best network were tested for each node in the simulation, the impact of the processor load was minimal. ESN’s lightweight processing was the main unexpected discovery in this experiment.

5.4.3. *Experiment 3 - comparison with other approaches*

In this experiment, we compared the performance of our engine with some “as is” approaches from opportunistic networks. The same scenario setup was employed as in the previous experiments. The protocols used for purposes of comparison were Prophet, DRAFT, Spray&Wait and BubbleRap. These were chosen because they represent different classes of “intelligent” protocols and have been extensively studied by researchers [33,74,75]. The following configuration parameters were employed: in Prophet - secondsInTimeUnit = 30; In DRAFT - familiarThreshold = 120, degrade = 0.5 and frame size = 3600; In Bubblerap K = 5, familiarThreshold = 700, centralityTimeWindow= 3600 and epoch/count = 6 (i.e. simulation time of 21,600 s/centralityTimeWindow 3600 = 6. Selecting parameters for protocols in this way, or “cherry picking”, is acceptable and consistent with the original paper of such approaches.

The results from the comparison of selected approaches in a scenario with 100 nodes are shown in Fig. 13.

As can be seen, the experiments with Spray&Wait, DRAFT and Prophet show similar results to our engine in number of delivered messages. Our approach is only inferior to Spray&Wait. However, it should be noted that in terms of the number of started and relayed messages and the overhead bandwidth ratio, our engine is almost 12% higher. This result can be attributed to the low number of started/relayed messages. The number of relayed messages was 632% minor in comparison with Spray&Wait. This rate indicates that our strategy to just relay messages to effective data mules generates low overhead, thus saving computational costs. In order to relay messages, mobile devices use battery power for processing and data transmission; thus, if this overhead indicator has a high value, it can influence the battery life of these devices.

BubbleRap had the lowest number of delivered messages. BubbleRap uses social metrics to make the forwarding decision. We believe that such low value is due to sparse contacts of nodes not characterizing communities of nodes. Despite the number of delivered messages sent by BubbleRap, DRAFT and Prophet generated significant overhead in comparison with our engine. These initiatives have an acceptable performance in some opportunistic networks scenarios with similar characteristics (e.g. social events, campus, school, manufactory) but as we had suspected, when it was used in an urban sensing scenario, it was found to be unsuitable due to high dynamicity of nodes, lack of communities (a feature used by BubbleRap), clusters (as explored by DRAFT) or frequent encounters (explored by Prophet).

DRAFT had an impressive number of started and relayed messages resulting in 900% more overhead than Prophet (second worst overhead rate). DRAFT combines spatial clustering with a decay function to create dynamic encounter graphs. This means that clusters reflect current and recent behaviour patterns by excluding devices which have not been seen for a long time [33]. Thus, this overhead rate indicates that DRAFT was having difficulties forming satisfactory clusters due to sparsity of encounters. With smaller clusters, it was failing to deliver packets to their final destinations [76]. To mitigate this issue, we tried to fine tune their configurations, changing values of degrade of decay ranging from 0.5 to 0.8; the time frame value was tested with values of 3600, 1800 and 7200; the familiar threshold was set with values of 120, 60 and 240 seconds. But even changing such values in different ex-

[7] http://www.cs.waikato.ac.nz/~ml/weka/.
[8] http://www.mathworks.com/.
[9] https://code.google.com/p/matlabcontrol/.

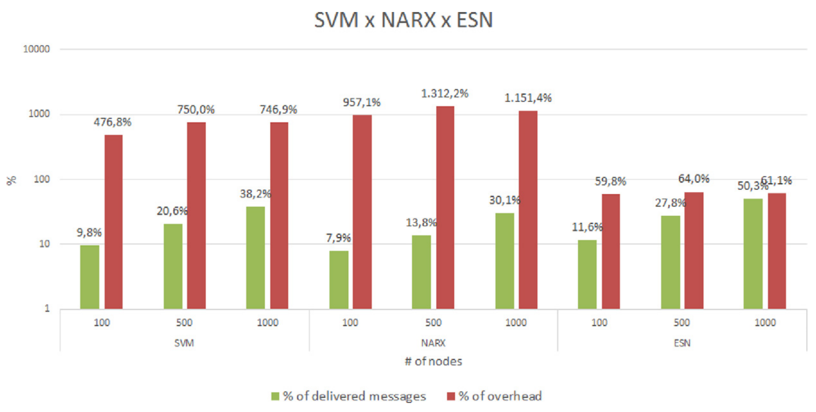
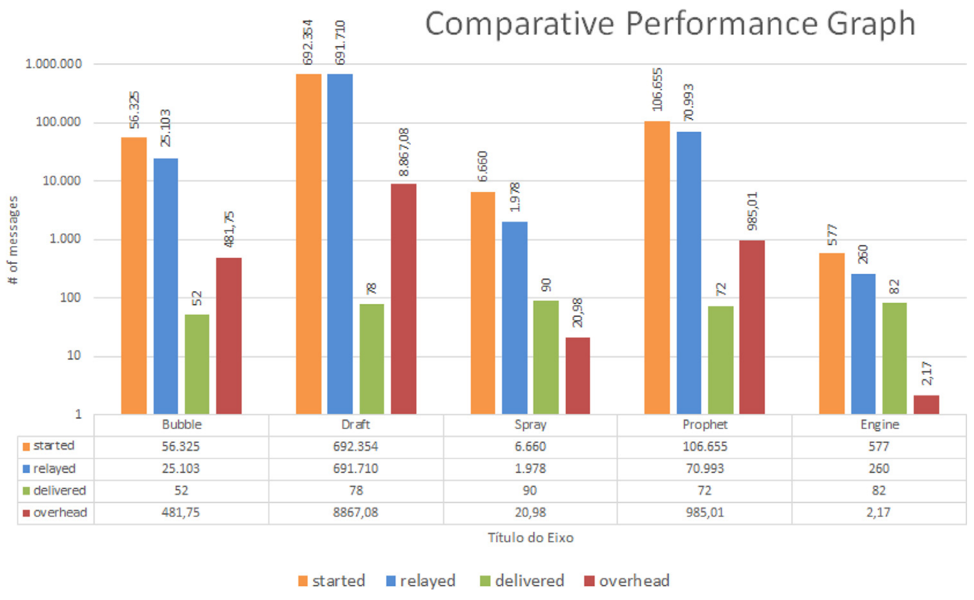


Fig. 12. Comparison of ESN with other neural networks.



	Bubble	Draft	Spray	Prophet	Engine
🟧 started	56.325	692.354	6.660	106.655	577
🟦 relayed	25.103	691.710	1.978	70.993	260
🟩 delivered	52	78	90	72	82
🟥 overhead	481,75	8867,08	20,98	985,01	2,17

Fig. 13. Comparison based on opportunistic networks approaches.

periments, the values of delivered messages did not change and overhead had few changes. Therefore, as indicated by the authors of DRAFT, the random message generation, short TTL of messages and fragmented nature of scenario are interfering in results.

On the whole, the results suggest that our engine achieved a satisfactory performance in terms of the number of delivered messages and overhead ratio and could be used in large-scale urban scenarios where the network infrastructure is intermittent or unavailable, such as in smart cities.

6. Conclusion and suggestions for future work

In this paper, we have described our attempt to build an engine that employs the opportunistic networks paradigm to transmit sensed data in situations where the networking infrastructure is intermittent or unavailable. Additionally, our experiments applied situation awareness and computational intelligence approaches to make decisions about the routing of messages and adaptation decisions. The proposed engine will be used as a basis for the data transmission in the Communication component of a large-scale architecture called UrboSenti. We have also outlined our design models for the software modules and their internal components. Currently, we are working on the incorporation of new dynamic rules in Situation Manager. The preliminary results obtained from a statistical situation set are acceptable. We believe that the performance of engine will be improved when such implementation is finished. The experiments also showed that ESN is a reliable technique for prediction. It achieved an impressive predictive performance and has a low computational cost compared with all the other approaches that we had previously adopted. In addition, the results revealed that some popular opportunistic networks initiatives cannot be used “as is” in the area of urban sensing applications.

Finally, the proposed engine is able to fill the gap of data transmission that was outlined 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.

In 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 embedded and intend to explore the application of a Deep Belief Network (DBN) or Restricted Boltzmann machines (RBMs) for the purposes of prediction.

References

[1] M. Mordacchini, L. Valerio, M. Conti, A. Passarella, Design and evaluation of a cognitive approach for disseminating semantic knowledge and content in opportunistic networks, Comput. Commun. 81 (2016) 12–30, doi:10.1016/j.comcom.2015.09.027.

[2] A. Zanni, Cyber-Physical Systems and Smart Cities, Technical Report, IBM Corporation, 2015.
[3] S. Pellicer, G. Santa, A.L. Bleda, R. Maestre, A.J. Jara, A.G. Skarmeta, A global perspective of smart cities: a survey, 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (2013) 439–444. 10.1109/IMIS.2013.79.
[4] B. Jedari, F. Xia, A survey on routing and data dissemination in opportunistic mobile social networks, IEEE Commun. Surv. Tut. (2013).
[5] M. Conti, S.K. Das, C. Bisdikian, M. Kumar, L.M. Ni, A. Passarella, G. Roussos, G. Tröster, G. Tsudik, F. Zambonelli, Looking ahead in pervasive computing: challenges and opportunities in the era of cyberphysical convergence, Pervasive Mob. Comput. 8 (1) (2012) 2–21, doi:10.1016/j.pmcj.2011.10.001.
[6] C.O. Rolim, A.G. Rossetto, V.R.Q. Leithardt, G.A. Borges, T.F.M. dos Santos, A.M. Souza, C.F. Geyer, An ubiquitous service-oriented architecture for urban sensing, in: F. Koch, F. Meneguzzi, K. Lakkaraju (Eds.), Agent Technology for Intelligent Mobile Services and Smart Societies, Communications in Computer and Information Science, 498, Springer Berlin Heidelberg, 2015, pp. 1–10, doi:10.1007/978-3-662-46241-6_1.
[7] I. Celino, S. Kotoulas, Smart cities [guest editors' introduction], IEEE Internet Comput. 17 (6) (2013) 8–11, doi:10.1109/MIC.2013.117.
[8] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, Smart cities and the future internet : towards cooperation frameworks for open innovation, Future Internet Lect. Notes Comput. Sci. 6656 (2011) 431–446, doi:10.1007/978-3-642-20898-0_31.
[9] M. Faschang, F. Judex, A. Schuster, Functional view of a smart city architecture: the SCDA greenfield approach, in: 2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 53, IEEE, 2016, pp. 1–6, doi:10.1109/MSCPES.2016.7480219.
[10] S.B. Eisenman, N.D. Lane, E. Miluzzo, R.A. Peterson, G.-s. Ahn, A.T. Campbell, MetroSense project: people-centric sensing at scale, in: Proc. of Workshop on World-Sensor-Web (WSW 2006), Boulder, 2006, pp. 6–11.
[11] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, N. Triandopoulos, AnonySense: a system for anonymous opportunistic sensing, Pervasive Mob. Comput. 7 (1) (2011) 16–30, doi:10.1016/j.pmcj.2010.04.001.
[12] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, N. Triandopoulos, AnonySense : privacy-aware people-centric sensing categories and subject descriptors, in: MobiSys'08 - Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services, Colorado, USA, 2008. 978-1-60558-139-2/08/06.
[13] M.-r. Ra, B. Liu, T.F. La Porta, R. Govindan, Medusa: a programming framework for crowd-sensing applications, in: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services - MobiSys '12, Section 2, ACM Press, New York, New York, USA, 2012, p. 337, doi:10.1145/2307636.2307668.
[14] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, R. Curtmola, Fostering participaction in smart cities: a geo-social crowdsensing platform, IEEE Commun. Mag. 51 (6) (2013) 112–119, doi:10.1109/MCOM.2013.6525603.
[15] T. Das, P. Mohan, V.N. Padmanabhan, R. Ramjee, A. Sharma, PRISM: platform for remote sensing using smartphones, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services - MobiSys '10, ACM Press, New York, New York, USA, 2010, pp. 63–76, doi:10.1145/1814433.1814442.
[16] P. Wu, J. Zhu, J.Y. Zhang, MobiSens: a versatile mobile sensing Platform for real-world applications, Mob. Netw. Appl. 18 (1) (2012) 60–80, doi:10.1007/s11036-012-0422-y.
[17] N. Brouwers, K. Langendoen, Pogo, a middleware for mobile phone sensing, in: P. Narasimhan, P. Triantafillou (Eds.), ACM/IFIP/USENIX 13th International Middleware Conference, Montreal, QC, Canada, December 3–7, 2012. Proceedings, Springer Berlin Heidelberg, 2012, pp. 21–40, doi:10.1007/978-3-642-35170-9_2.
[18] E. Miluzzo, N.D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S.B. Eisenman, X. Zheng, A.T. Campbell, Sensing meets mobile social networks : the design , implementation and evaluation of the CenceMe application, SenSys '08 (2008) 337–350, doi:10.1145/1460412.1460445.
[19] E. Theodoridis, G. Mylonas, V. Gutiérrez, L. Muñoz, Large-scale participatory sensing experimentation using smartphones within a smart city, in: Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2014, pp. 178–187, doi:10.4108/icst.mobiquitous.2014.258016.
[20] H.A. Nguyen, S. Giordano, Routing in opportunistic networks, Int. J. Amb. Comput. Intell. 1 (3) (2009) 19–38, doi:10.4018/jaci.2009070102.
[21] M. Conti, M. Kumar, Opportunities in opportunistic computing, Computer 43 (1) (2010) 42–50, doi:10.1109/MC.2010.19.
[22] W. Ivancic, W. Eddy, D.C. Iannicca, J. Ishac, A.G. Hylton, Store, carry and forward problem statement, Internet-Draft, Internet Engineering Task Force, 2014. draft-ivancic-scf-problem-statement-01, Work in Progress
[23] H.A. Nguyen, S. Giordano, Context information prediction for social-based routing in opportunistic networks, Ad Hoc Netw. 10 (8) (2012) 1557–1569, doi:10.1016/j.adhoc.2011.05.007.
[24] A. Vahdat, D. Becker, Epidemic Routing for Partially-Connected Ad Hoc Networks, Technical Report, Duke University, 2000.
[25] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Spray and wait, in: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking - WDTN '05, ACM Press, New York, New York, USA, 2005, pp. 252–259, doi:10.1145/1080139.1080143.
[26] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Efficient routing in intermittently connected mobile networks: the multiple-copy case, IEEE/ACM Trans. Netw. 16 (2008) 77–90, doi:10.1109/TNET.2007.897964.
[27] A. Mathurapoj, C. Pornavalai, G. Chakraborty, Fuzzy-Spray: Efficient routing in delay tolerant ad-hoc network based on fuzzy decision mechanism, 2009 IEEE Int. Conf. Fuzzy Syst. (2009) 104–109, doi:10.1109/FUZZY.2009.5277223.
[28] A. Lindgren, A. Doria, O. Schelén, Probabilistic routing in intermittently connected networks, ACM SIGMOBILE Mob. Comput. Commun. Rev. 7 (3) (2003) 19, doi:10.1145/961268.961272.
[29] P. Hui, J. Crowcroft, E. Yoneki, BUBBLE rap: social-based forwarding in Delay-Tolerant networks, IEEE Trans. Mob. Comput. 10 (11) (2011) 1576–1589, doi:10.1109/TMC.2010.246.
[30] W. Moreira, P. Mendes, S. Sargento, Opportunistic routing based on daily routines, in: 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE, 2012, pp. 1–6, doi:10.1109/WoWMoM.2012.6263749.
[31] R.I. Ciobanu, C. Dobre, V. Cristea, SPRINT: social prediction-based opportunistic routing, 2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2013(2013). 10.1109/WoWMoM.2013.6583442.
[32] P. Nabhani, A. Masoud Bidgoli, Adaptive fuzzy routing in opportunistic network (AFRON), Int. J. Comput. Appl. 52 (18) (2012) 7–11, doi:10.5120/8299-1520.
[33] M. Orlinski, N. Filer, The rise and fall of spatio-temporal clusters in mobile ad hoc networks, Ad Hoc Netw. (2013), doi:10.1016/j.adhoc.2013.03.003.
[34] E.C.R. de Oliveira, C.V.N. de Albuquerque, CARTOON - context aware routing over opportunistic netowkrs, in: Proc. XXX Brazilian Symposium on Computer Networks and Distributed Systems, Ouro Preto, MG, Brazil, 2012, pp. 872–885.
[35] M. Musolesi, Context-aware Adaptive Routing for Delay Tolerant Networking, University of London, 2007 Phd thesis.
[36] C. Boldrini, M. Conti, I. Iacopini, A. Passarella, HiBOp: a history based routing protocol for opportunistic networks, 2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WOWMOM, 2007, doi:10.1109/WOWMOM.2007.4351716.
[37] A. Balasubramanian, B.N. Levine, A. Venkataramani, Replication routing in DTNs: a resource allocation approach, IEEE/ACM Trans. Netw. 18 (2) (2010) 596–609, doi:10.1109/TNET.2009.2036365.
[38] H.A. Nguyen, S. Giordano, PROSAN: probabilistic opportunistic routing in SANETs, in: Proceedings of the First ACM workshop on Sensor and actor networks - SANET '07, ACM Press, New York, New York, USA, 2007, p. 11, doi:10.1145/1287731.1287735.
[39] W.M. Moreira, P. Mendes, S. Sargento, Social-aware opportunistic routing protocol based on user's interactions and interests, Ad Hoc Netw. 129 (2014), doi:10.1007/978-3-319-04105-6_7.
[40] C. Boldrini, J.Y.L. Boudec, A. Chaintreau, M. Conti, Deliverable 2.2: Final specification of Forwarding Paradigms in Haggle, Technical Report, 2008.
[41] M. Conti, C. Boldrini, S.S. Kanhere, E. Mingozzi, E. Pagani, P.M. Ruiz, M. Younis, From MANET to people-centric networking: milestones and open research challenges, Comput. Commun. 000 (2015) 1–21, doi:10.1016/j.comcom.2015.09.007.
[42] M. Musolesi, C. Mascolo, CAR: context-aware adaptive routing for delay-tolerant mobile networks, IEEE Trans. Mob. Comput. 8 (2) (2009) 246–260, doi:10.1109/TMC.2008.107.
[43] J. Ye, S. Dobson, S. McKeever, A review of situation identification techniques in pervasive computing, Pervasive Mob. Comput. In Press, (0) (2011), doi:10.1016/j.pmcj.2011.01.004.
[44] M.R. Endsley, Toward a Theory of Situation Awareness in Dynamic Systems, 1995.
[45] S.S. Yau, F. Karim, S.K.S. Gupta, Reconfigurable context-sensitive middleware for pervasive computing, IEEE Pervasive Comput. 1 (3) (2002) 33–40, doi:10.1109/MPRV.2002.1037720.
[46] A. Manzalini, F. Zambonelli, Towards autonomic and situation-aware communication services: the CASCADAS vision, Proc. IEEE (1) (2006) 2–7.
[47] B. Craenen, A. Eiben, Computational intelligence, in: Encyclopedia of Life Support Sciences (EOLSS)., EOLSS Publishers Co. Ltd, 2009, p. 142.
[48] J.C. Bezdek, What is computational intelligence?, in: J.M. Zurada, R.J. Marks II, C.J. Robinson (Eds.) Computational Intelligence, Imitating Life, IEEE Computer Society Press, Piscataway, 1994, pp. 1–12.
[49] M.R. Przybylek, Computational Intelligence 465(2013) 119–134. 10.1007/978-3-642-35638-4.
[50] P. Trebatický, Prediction of dynamical systems by recurrent neural networks, Inf. Sci Technol. Bull. ACHM Slovakia 1 (1) (2009) 47–56.
[51] Z. Xiang, X. Deyun, Fault diagnosis based on the fuzzy-recurrent neural networks, Asian J. Control 3 (2) (2001) 89–95.
[52] M. Lukosevicius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comput. Sci. Rev. Preprint submitted. (2010).
[53] H. Jaeger, 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, 2001.
[54] W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: a new framework for neural computation based on perturbations., Neural Comput. 14 (11) (2002) 2531–2560, doi:10.1162/089976602760407955.
[55] S. Wang, T. Chen, X. Xu, Flue gas turbine condition trend prediction based on improved echo state network, in: Proceedings - 3rd International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2011, 2, 2011, pp. 242–245, doi:10.1109/ICMTMA.2011.348.

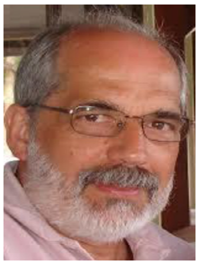
[56] S. Morando, S. Jemei, R. Gouriveau, N. Zerhouni, D. Hissel, Fuel cells remaining useful lifetime forecasting using echo state network, in: 2014 IEEE Vehicle Power and Propulsion Conference (VPPC), 2014, pp. 1–6, doi:10.1109/VPPC.2014.7007074.
[57] M.J.A. Rabin, M.S. Hossain, M.S. Ahsan, M.A.S. Mollah, M.T. Rahman, Sensitivity learning oriented nonmonotonic multi reservoir echo state network for short-term load forecasting, in: 2013 International Conference on Informatics, Electronics and Vision (ICIEV), IEEE, 2013, pp. 1–6, doi:10.1109/ICIEV.2013.6572692.
[58] M.J.A. Rabin, M.S. Hossain, M.S. Ahsan, M.A.S. Mollah, a. N. M. E. Kabir, M. Shahjahan, Electrical load forecasting using echo state network, 2012 15th International Conference on Computer and Information Technology (ICCIT) (2012) 50–54. 10.1109/ICCITechn.2012.6509763.
[59] X.-l. Xu, T. Chen, S.-h. Wang, Condition prediction of flue gas turbine based on Echo State Network, in: 2010 Sixth International Conference on Natural Computation, IEEE, 2010, pp. 1089–1092, doi:10.1109/ICNC.2010.5583012.
[60] R.R.B. de Aquino, O.N. Neto, R.B. Souza, M.M.S. Lira, M.A. Carvalho, T.B. Ludermir, A.A. Ferreira, Investigating the Use of Echo State Networks for Prediction of Wind Power Generation, in: 2014 IEEE Symposium on Computational Intelligence for Engineering Solutions (CIES), 2014, pp. 148–154, doi:10.1109/CIES.2014.7011844.
[61] L. Maciel, F. Gomide, D. Santos, R. Ballini, Exchange rate forecasting using echo state networks for trading strategies, in: 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), 2014, pp. 40–47, doi:10.1109/CIFEr.2014.6924052.
[62] P. Yu, L. Miao, G. Jia, Clustered complex echo state networks for traffic forecasting with prior knowledge, in: Conference Record - IEEE Instrumentation and Measurement Technology Conference, 2011, pp. 670–674, doi:10.1109/IMTC.2011.5944078.
[63] J.-S. Jang, Neuro-fuzzy modeling and control, Proc. IEEE 83 (3) (1995) 378–406, doi:10.1109/5.364486.
[64] L.a. Zadeh, Is there a need for fuzzy logic? Inf. Sci. 178 (13) (2008) 2751–2779, doi:10.1016/j.ins.2008.02.012.
[65] J.-S. R. Jang, C.-T. Sun, Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.
[66] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, An adaptive location prediction model based on fuzzy control, Comput. Commun. 34 (7) (2011) 816–834, doi:10.1016/j.comcom.2010.09.001.
[67] M. Perttunen, J. Riekki, O. Lassila, Context representation and reasoning in pervasive computing: a review, Int. J. Multimedia Ubiquitous Eng. 4 (4) (2009) 1–28.
[68] F. Li, Y. Yang, J. Wu, X. Zou, Fuzzy closeness-based delegation forwarding in delay tolerant networks, Proceedings - 2010 IEEE International Conference on Networking, Architecture and Storage, NAS 2010 (2010) 333–340. 10.1109/NAS.2010.65.
[69] J. Makhlouta, H. Harkous, F. Hutayt, H. Artail, Adaptive fuzzy spray and wait: efficient routing for opportunistic networks, in: 2011 International Conference on Selected Topics in Mobile and Wireless Networking (iCOST), IEEE, 2011, pp. 64–69, doi:10.1109/iCOST.2011.6085837.
[70] P. Bellavista, Pervasive computing at scale: challenges and research directions, in: 2011 IEEE SENSORS Proceedings, IEEE, 2011, pp. 639–642, doi:10.1109/ICSENS.2011.6127000.
[71] D. Rodrigues-silva, A. Costa, J. Macedo, Energy impact analysis on DTN routing protocols, ExtremeCom' 12, Zurich, Switzerland, 2012.
[72] D. Karamshuk, C. Boldrini, M. Conti, A. Passarella, Human mobility models for opportunistic networks, IEEE Commun. Mag. 49 (12) (2011) 157–165, doi:10.1109/MCOM.2011.6094021.
[73] C.O. Rolim, A.G. Rossetto, V.R. Leithardt, G.A. Borges, T.F. dos Santos, A.M. Souza, C.F. Geyer, Towards predictive routing agents in opportunistic networks, in: Fifth International Workshop on Collaborative Agents Research & Development, CARE for Intelligent Mobile Services - Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Springer-Verlag Berlin Heidelberg, Paris, France, 2014.
[74] A. Bujari, A survey of opportunistic data gathering and dissemination techniques, 2012 21st International Conference on Computer Communications and Networks (ICCCN) (2012) 1–6. 10.1109/ICCCN.2012.6289225
[75] C. Boldrini, M. Conti, F. Delmastro, A. Passarella, Context- and social-aware middleware for opportunistic networks, J. Netw. Comput. Appl. 33 (5) (2010) 525–541, doi:10.1016/j.jnca.2010.03.017.
[76] M. Orlinski, N. Filer, Quality distributed community formation for data delivery in pocket switched networks, in: Proceedings of the Fourth Annual Workshop on Simplifying Complex Networks for Practitioners - SIMPLEX '12, ACM Press, New York, New York, USA, 2012, p. 31, doi:10.1145/2184356.2184365.

Carlos O. Rolim. Ph.D. in computing at the Federal University of Rio Grande do Sul (2011) under the supervision of Prof. Dr. Cláudio Fernando Resin Geyer, a master's degree in computer science from the Federal University of Santa Catarina (2007) and a degree in computer science (2003). He is member of Parallel Processing and Distributed Group (GPPD) - UFRGS/II and associate researches at Networks and Management Laboratory (LRG) - UFSC / CTC / INE. He has experience in Networks and Distributed Systems area. His research interest are: ubiquitous and pervasive computing, opportunistic networks, smart cities, grid computing and cloud computing.

Anubis G. M Rossetto. graduate in computer science at the University of Passo Fundo (1998) and master's degree in computer science at Federal University of Santa Catarina (2007). Is a professor at the Federal Institute of Rio Grande do Sul. Areas of concern: web applications, mobile computing, ubiquitous computing, fault tolerance. Currently, Ph.D. student in computer science at the Federal University of Rio Grande do Sul.

Valderi R. Q. Leithardt. bachelor's at technology from Data Processing Higher Education Center of Foz do Iguaçu (2002) and master's at computer science from the Catholic University of Rio Grande do Sul (2008). Has experience in computer science, focusing on Computer Systems Architecture, acting on the Following subjects: ubiquitous computing, pervasive computing, information systems, computer accessibility, privacy and technical course.

Guilherme A. Borges. graduated in technology systems to Internet by the Federal Institute South - Rio-Grande (2012). The area of expertise is in distributed systems with interest in systems for web, embedded systems, ubiquitous computing and mobile computing. He is currently a master's student in computing the Federal University of Rio Grande do Sul.



Claudio F. R. Geyer. graduate at mechanical engineering from the Federal University of Rio Grande do Sul (1978), master's at computer science from the Federal University of Rio Grande do Sul (1986) and Ph.D. at Informatica from Universite Grenoble I (Scientifique Et Medicale - Joseph Fourier) (1991). Has experience in computer science, focusing on computer systems, acting on the Following subjects: pervasive computing, grid/cloud/volunteer computing, Big Data systems and massive multiplayer games.

Tatiana F. Mousquer dos Santos. graduated in pedagogy (2010) and information systems (ongoing), master's in Production Engineering at Federal University of Santa Maria (2014). Her research interest are statistical and mathematical approaches in computer science with focus in ubicomp and grid computing area.

Adriano M. Souza. graduated in mathematics, specialization in statistics and quantitative modeling and master in production engineering (UFSM). Ph.D. in production engineering in Federal University of Santa Catarina (2000). He is currently associate professor, in Department of Statistics (UFSM) acting in the course of Specialization in statistics and quantitative modeling (UFSM) and master in production engineering (UFSM) is part of the editorial board of the journal Science and Natura. The area of interest are applied statistics, quality control, time series analysis, multivariate analysis, operations research and management.