Echo State Networks and Fuzzy Logic to Improve Data Dissemination in Opportunistic Networks

Carlos O. Rolim, Valderi R. Q. Leithardt, Anubis G. Rossetto, Guilherme Borges, Claudio F. R. Geyer
Institute of Informatics
Federal University of Rio Grande do Sul (UFRGS)
{carlos.oberdan, valderi.quietinho, agmrossetto,gaborges, geyer}@inf.ufrgs.br

Tatiana F. M. dos Santos, Adriano M. Souza
Postgraduate Program in Production Engineering
Federal University of Santa Maria (UFSM)
taty.nanda@gmail.com, amsouza@ufsm.br

Abstract

Opportunistic Networks uses pair-wise contacts to share and forward content without any prior knowledge of the pre-existing infrastructure. In this context, optimizing data dissemination among nodes is a paramount requirement. Current solutions are often based on instantaneous contextual information for routing decisions that do not take account of a possible future situation. In this paper, we present a new approach of a smarter engine that exploits the prediction of future values of context to improve routing in Opportunistic Networks. It uses Echo State Networks (ESN) for prediction and Fuzzy Logic for decision making process. We carried out simulations using an environment simulator and achieved positive results.

1. Introduction

The new field of mobile application gives rise to several challenges. One of these concerns is the use of mobile computing as an underlying technology that can supply collaborative sensing data and social networking metadata for these ubiquitous services in the context of Smart Cities. In these wide-scale urban scenarios, just relying on wireless infrastructures (e.g. cellular, WLAN, or WiMAX networks) to provide services is not satisfactory, as it is very unlikely that wireless infrastructures alone will be able to provide enough bandwidth and coverage for the huge number of devices spread throughout the environment [1]. Another important consideration is that current mobile computing applications are infrastructure-centric; this forces users to be acutely aware of their connectivity environment, since there are many applications that can only work when there is a networking infra-structure available.

An alternative way of overcoming these limitations is the use of Opportunistic Networks. Opportunistic Networks are a recent mobile networking paradigm that stem from research into conventional Mobile Ad Hoc NET-works (MANET). In this paradigm, the nodes are assumed to be mobile, and the forwarding of messages is based on the "Store, Carry and Forward" (SCF) concept. Opportunistic Networks rep-resent the first attempt to close the gap between human and network behavior 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.

One fundamental problem in this context is how to route messages from the source node 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 whole life-time of the message. Thus, in this paper a new approach is outlined that involves a smarter engine for routing in Opportunistic Networks. This employs Echo State Networks (ESN) and Fuzzy Logic to make inferences about context data, predict future values and make the most suitable routing decision. Our study is based on the hypothesis that since each node has some contextual data that describes its current situation, if it was possible to predict future values, we could infer the future situation of the node and thus improve routing. In summary, the contributions made by this paper are as follows: first of all, this is the first paper to apply a lightweight neural network called Echo State Network in the area of Opportunistic Networking. Second we provide a novel engine architecture that deals with prediction and uncertainty with acceptable results. Third, this

study extends our previous work [4] [2] [3]. As well as being original, this covers all the research findings made by us in this area. The rest of this paper is structured as follows: The next section describes our proposed smarter engine architecture; Section 3 describes our experiments and analyzes the results; and, finally we conclude this paper in Section 4.

2. Proposed Architecture

As shown above, a major problem in Opportunistic Networks is how to route messages between the nodes without having a previous knowledge of the network topology. To address this problem, we propose a new approach involving a smarter engine. Each node of the network runs this embedded engine. Its main task is to collect current context data, predict future values of context parameters and make decisions about routing. In this study, we have incorporated knowledge obtained from previous results to choose a conceptually simple and computationally inexpensive kind of neural network called Echo State Network. This is being employed as the underlying technique for the Forecaster module and Fuzzy Logic to deal with any uncertainty arising from the current and future context data in the Decision Making module. The architecture of our smarter engine is outlined in Figure 1 and its behavior is explained below.

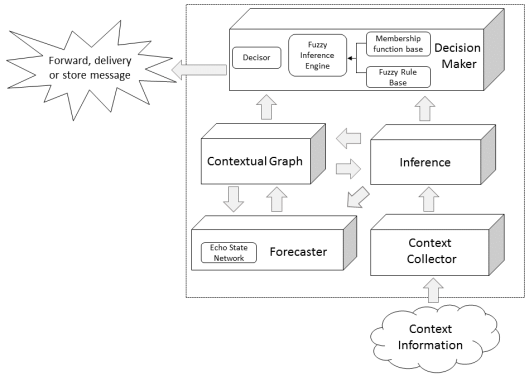

Figure 1. Smarter Engine architecture.

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 *Contextual Graph*, thus creating a new Layer 2 vertex. Contextual graph underlies all data storage. It is a graph structure with vertices arranged in layers representing different kinds of information.

The *Inference* module, use a set of internal variables to make inferences about node situations. It draws on data collected by the Context Collector and data stored in the Contextual Graph for this purpose. The information generated by the Inference module will be used later by the Decision Maker. It also runs maintenance routines like pruning old data and indicating when the Context Collector and Forecaster should be run.

The *Forecaster* module implements an Echo State Network (ESN) for prediction. ESN are a kind of three-layered recurrent network with sparse, random, and (crucially), untrained connections within the recurrent hidden layer. When the *Inference* module detects a sufficient amount of context information, the *Forecaster* module is invoked. The Forecaster then attempts to find the most suitable network configuration depending on the node situation. In the training phase, the historical context data of the nodes is "pumped" to the internal structures so that it can be used as input to train the network. The internal global parameters of ESN (i.e. size of reservoir, sparsity of the reservoir, spectral radius and leaking rate) are set and the network starts to run by attempting to find a network configuration with a lower Mean Squared Error (MSE). At every iteration, these values are changed and the best network configuration is stored (on average 810 different configurations are tested). After completing the training, the best network is saved and is ready to predict future values in the exploitation phase. During exploitation phase, the structure with historical values is used to "pump" the best network (found and saved in previous phase) with some data steps and thus to activate the internal reservoir. Some stages later, the input from the historical data is switched off to allow the network to predict values by itself. There is a problem with regard to the question of how to stop providing input because the network adapts to the continuous input stream during the training phase. This is carried out by adjusting output weights via a multiple linear regression which considers the input layer. If this input is switched off during the exploitation phase, the reservoir activity will decline quite fast (depending on the spectral radius and output feedback weights) and the network cannot respond in the way it was trained to anymore. A solution that was found was to use initial steps to activate the reservoir and when the input from the historical data is switched off, the network itself predicts the output that will be used as the next input step in a self-recurrent way. The predicted values are stored at *Contextual Graph* as Layer 3 vertices. At this point, the *Inference* module sets an internal variable to show that the node is now running in smart mode and *Decision Maker* module could use the current context data with these predicted data to improve routing decision.

Decision Maker runs when a message has to leave the buffer in the node (due to a contact with another node). This module decides if this 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 ca-

pacity 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, we applied Fuzzy Logic. The internal functions of *Decision Maker* retrieve the all context values (current, historical and predicted) of the current and contacted node from *Contextual Graph* and use it as input for the Fuzzy Inference System (FIS). The FIS uses internal components to calculate the potential of each node. The potential values of nodes are used by *Decisor* in decision making process.

3. Simulation and Experimental results

3.1. Simulation setup

Our simulations were carried out by using a ONE (Opportunistic Network Environment) Simulator. We chose a timescale of 21,600 seconds = 6 hours for all the scenarios. Each simulation scenario was run with a different number of nodes (10 for the first, 25 for the second, and 50, 75 and 100 for each subsequent group). Two groups were used: pedestrians and cars. The pedestrian nodes moved between 0.5 and 1.5 Km/h, and had a Bluetooth device with a radio range of 20 meters and transmission speed of 2 Mbit/s. The Car nodes moved between 10 and 50 Km/h and had a Wi-Fi interface with a range of 50 meters and transmission speed of 10 Mbit/s. On average, the nodes generated about one message every 25 to 35 seconds (total of 711) in all of the experiments and the message lifetime was set at 24 minutes (1440 seconds). Some energy constraints were imposed on all the nodes during the simulation. All the nodes started with a battery charge of 2400, with the battery being recharged every 43,200 seconds. The energy expenditure of other nodes by scan was set at 0.92 per second and the energy expenditure to send and receive a message was set at 0.08 per second. The context information of each node was collected by our engine at intervals of 100 seconds.

The ESN was built using ESNJava software. It provides a graphical interface which makes it easier to handle ESN networks and an API to embed ESN in Java applications. However, the ESNJava just handles situations where a sequence of values that must be learned (teacher-forced) 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 beat by making changes in the source code of ESNJava. The original code of ESNJava was changed to introduce support for the network itself predicts the output that will be used as the next input step in a self-recurrent way (as introduced in description of *Forecaster* module at section 2). In contrast of approaches presented in some of our previous work which used a fixed configuration for all the nodes, in this implementation of our smarter engine, each node of the network builds its own ESN with the most appropriate configuration for its context. This is carried out by testing different network configuration until the best one (i.e. the one with minimal MSE) is found. The Figure 2 graphically represent this process indicating that for such node the best network configuration had network size = 10 and spectral radius = 0.77. As each node executes this process, 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.

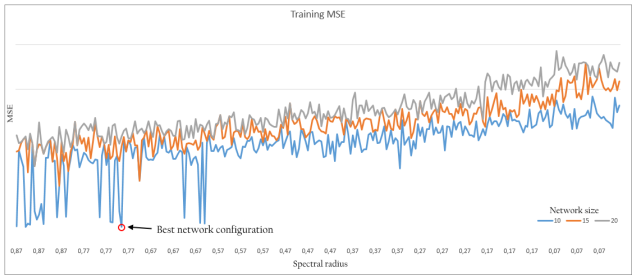

Figure 2. Selection of best network configuration

The Fuzzy Inference System (FIS) used in the *Decision Maker module* was implemented using JFuzzyLogic library. 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.

3.2. Experimental results

We conducted a set of experiments using the simulation setup outlined in Section 3.1. The purpose of our experiments was to measure the performance of our smarter engine by increasing the number of nodes in the network. We are seeking to test the impact of the number of nodes on the number of messages created, started, relayed and delivered, as well as on the overhead ratio of the bandwidth. The results are shown in Figure 3.

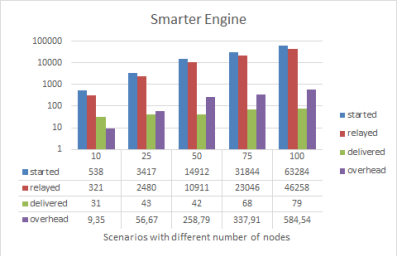

Figure 3. Performance of the smarter engine with a different number of nodes

As expected, the number of delivered messages increases with the increment of the number of nodes. In the scenario with 100 nodes, the total of 79 messages is a good amount, compared with some other popular protocols used in Opportunistic Networks. One factor that drew our attention was the number of started and relayed messages. We believe that this can be attributed to the fact we are only removing messages from the local buffer when the time to life (TTL) of messages has been reached and not when a message is forwarded. Hence, the same message is being forwarded to several nodes. This conservative strategy of buffer management should be reviewed because, as well as generating more traffic in the network, it requires a greater expenditure of power from the nodes. However, despite the high overhead, this amount can be considered acceptable. One factor not reported in the chart is the computational cost of ESN. Even when each node used in the simulation testing, had, on average, 810 different configurations to find the best network, the impact of the processor load was minimal. We believe that its lightweight feature was the main differential of ESN when compared with all the other machine learning approaches that we have tested in our previous work. On the basis of these results it can be concluded that although we need to improve buffer management to reduce the impact on the overhead metrics, our smart engine had a satisfactory performance in terms of the number of delivered messages and could be used in wide-scale urban scenarios where network infrastructure is intermittent or unavailable, such as Smart Cities.

4. Conclusions

This paper has outlined a new approach to improve routing in Opportunistic Networks, using Echo State Networks (ESN) and Fuzzy Logic. Its main characteristic is the use of current and predicted context data for decision-making. We have successfully demonstrated the feasibility of integrating several techniques in one engine with satisfactory results since this led to a reasonable performance in terms of delivery messages. This suggests that our initial hypothesis to use prediction for better routing decisions has proved to be valid and provides strong grounds for undertaking further research. The experiments showed that ESN could be considered the best technique to be used as underlying for forecasting. In addition to achieving an impressive predictive performance, ESN has a low computational cost compared with all the other approaches that we already applied. On the basis of these findings, we argue that the proposed engine is suitable to be used in scenarios where the networking infrastructure is not always available, such as Smart Cities. Moreover, it should be noted that this is the first paper to demonstrate the feasibility of using ESN in Opportunistic Networks. 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.

References

[1] M. Conti, S. Giordano, M. May, and A. Passarella. From opportunistic networks to opportunistic computing. *IEEE Communications Magazine*, 48(9):126–139, Sept. 2010.

[2] C. O. Rolim, V. R. Leithardt, A. G. Rossetto, T. F. dos Santos, A. M. Souza, and C. F. Geyer. Six degrees of separation to improve routing in opportunistic networks. *International Journal of UbiComp*, 4(3):11–22, 2013.

[3] C. O. Rolim, V. R. Leithardt, A. Rossetto, G, and C. F. Geyer. Analysis of a hybrid neural network as underlying mechanism for a situation prediction engine. *Journal of Applied Computing Research*, 2(1):22–31, 2012.

[4] C. O. Rolim, A. G. Rossetto, V. R. Leithardt, G. A. Borges, T. F. dos Santos, A. M. Souza, and C. F. Geyer. Towards Predictive Routing Agents in Opportunistic Networks. In *5th Int Workshop on Collaborative Agents Research & Development (CARE)*, Paris, France, 2014. Springer-Verlag Berlin Heidelberg.