VOL. 16, NO. 17, SEPTEMBER 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

ERLANG B TRAFFIC TABLE MODELED AS AN ARTIFICIAL NEURONAL NETWORK IN AN APPLICATION FOR SMARTPHONE

Johan Julián Molina Mosquera, José de Jesús Salgado Patrón and Faiber Robayo Betancourt
Departamento de Ingeniería Electrónica, Facultad de Ingeniería, Universidad Surcolombiana, Neiva, Huila, Colombia
E-Mail: julian.molina@usco.edu.co

ABSTRACT

This paper presents the methodology to implement the non-linear function of Erlang B as a multilayer ANN (Artificial Neural Network). The number of channels and the traffic intensity of the system are used as input parameters, and the blocking probability or degree of service is used as output data. The proposed model allows the development of the view controller model of the native Android application, in which another variable is entered corresponding to the intensity of traffic per user. The Android app showed results with an error of 0.06%, storing the data to visualize them graphically, thus allowing to simulate and analyze the traffic of fixed telephony and mobile telephony systems.

Keywords: erlang B traffic table, artificial neural network, smartphone, traffic intensity, android app.

1. INTRODUCTION

Technologies seek new possibilities for the future of mobile communications. Provider companies present a portfolio with the latest network solutions, designed to increase the user experience as the intensity of traffic grows exponentially; Since the Erlang B card is widely used in telecommunications systems, it provides the reading of a maximum limit of 100 channels and up to 10% in the probability of blocking or GOS (Grade Of Service).

Therefore, the drawback of the Erlang B function is that it is not always practical in real applications, due to the computational complexity when using high rates of traffic intensity (A. Harrel, 1988) and a number of channels. For this reason, it is convenient to use approximate models and designs of artificial neural networks which respond better to the function non-linearities (Hagan Demuth, Beale *et al.*, 1996).

Since the beginning of the 20th century, there has been a rapid growth in the popularity of telecommunications systems. As the number of subscribers using telecommunications systems increases, the capacity increases in the number of calls made. Telecommunications systems are made up of expensive hardware devices, such as switches, with the function of carrying the traffic of telephone calls.

The traffic generated by phone calls is random because the number of calls generated within a period of time and the duration of each call are unpredictable. In telecommunications systems, over dimensioning the capacity requirements will lead to an economic problem due to high costs. On the other hand, by virtue of the dimensioning, the quality of service (QoS) will be reduced due to the degradation of the signal. Therefore, estimating the expected traffic generated is crucial in the design and dimensioning of telecommunications systems (Robert Cooper, Daniel Heyman, 1998), (Tibor Misuth, Ivan Baronak, 2011).

Teletraffic theory and Engineering deal with the problem of traffic and its relationships with other parameters of telecommunications systems (Villy Iversen, 2001) such as call congestion, congestion time, and traffic congestion. Currently, the Erlang B (Richard Parkinson, 2002) and Erlang C (William Lee, 1995) models are used to calculate the grade of service and call waiting probability respectively.

The Erlang B formula has a significant number of papers investigating the properties of the Erlang loss function (Eric Wong, Andrew Zalesky, Zvi Rosberg, and Moshe Zukerman, 2007.) (L. Takacs, 1969). Using this formula is not always practical. Sometimes, in real applications, it is more convenient to use approximations and computational intelligence (Angus, 2001) (Freeman, 2015).

The applications found on the Internet as a calculator or Erlang B function do not respond to a multilayer neural network design methodology. The calculations obtained are not very flexible and of reduced precision, determining factors in the optimization of resources in fixed and mobile communication systems.

In this way, in the research article, the Erlang B model will be taken as the main focus to design and train the multilayer artificial neural network, which according to the weights and bias obtained with the MATLAB software tool and the algorithm of Levenberg-Marquardt training, the native Android Studio development platform will be used to implement the MVC (Controller View Model) of the mobile application.

This document consists of the following basic sections: section II deals with the Methodology, in which the characteristics of the artificial neural network, characteristics of the ErlangB chart, and experimental configuration are addressed. For this configuration, the data is tabulated, the multilayer ANN is designed, the equations are obtained, the neural training is carried out, and the Android application MVC model is implemented. In the results analysis section, the functionality of the application is shown, and the data obtained from other existing applications and methods are compared. Finally, there are the sections of the data treatment, discussion, conclusions, and bibliographic reference.

2. METHODOLOGY

2.1 Artificial Neuronal Network Characteristics

An artificial neural network can be defined as a connectionist model whose elements or nodes, connected to each other, simulate the functions performed by brain cells, which are called neurons (Rafael Lahoz Beltrá, 2004).

They can also be defined as computer systems consisting of a large number of simple, highly interconnected elements that process information by responding dynamically to external stimuli.

An artificial neuron is a processing unit with four functional elements, as shown in Figure-1.

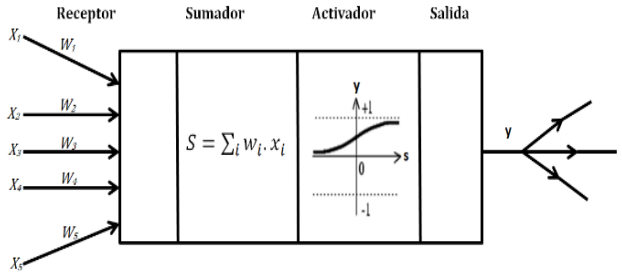


Figure-1. Structure of an artificial neuron.

The receptor receives one or more input signals Xi, which generally comes from other neurons and is each amplified or attenuated according to a weight factor Wi, which constitutes the connectivity between the source neuron from which they come and the neuron of destiny. On the other hand, the adder is in charge of carrying out the algebraic sum of the input signals, penetrating them according to their weight, applying the following expression:

$$S = \sum W_i X_i \quad (1)$$

Finally, the activation function applies a function to the simulator's output to decide whether the neuron is activated, triggering an output or not.

Table-1 shows the similarity between a natural biological neuron and its artificial representation:

Table-1. Analogies between biological and artificial neurons biological neuron artificial neuron.

Biological Neuron	Artificial Neuron
Signals reaching the synapses	Inputs to the neuron
Excitatory or inhibitory character of the input synapse	Input weights
Total stimulus of the neuron	$Net_j = \sum_{i=0}^{N} W_i(t).X_i(t)$
Activation or not of the neuron	Activation function
Neuron response	Output function

2.2 Characteristics of the Erlang b Chart

With the results obtained from the equation and with the guidance of the Erlang B chart, the data table is constructed taking as input variables the number of Channels "C" and the Traffic Intensity "A" (Erlangs), and as output variable, the probability of blocking.

$$P_{bloqueo} = \frac{\left(\frac{A^C}{C!}\right)}{\sum_{K=0}^{C}\frac{A^K}{K!}} = GOS \quad (2)$$

The function of equation (2) is a non-linear function; that is, it does not comply with the properties of addition and homogeneity.

Remembering that the linearity property is associated with the concept of vector space, a set in which two operations are defined, one internal (sum of vectors X + Y) and another external (multiplication by a scalar aX, in which a, belongs to an external set), hence the linearity property is expressed referring to these two operations.

As an example of the non-linearity in the ErlangB chart, for 5 channels, the traffic intensity varies between 0.7 and 2.7 Erlangs. When it is 6 channels, the variation of this intensity is between 1 and 3.5 Erlangs. The same thing is happening progressively for the other channels.

In this order of ideas, in order to accurately model the ANN of the Erlang B equation, it was necessary to design the data table and the multilayer neural network.

2.3 Experimental Configuration

2.3.1 Data tabulation

The data for each channel is obtained using the ErlangB chart and/or equation (2).

In Figure-2, the scatter plot of the Erlang B chart is shown up to 22 channels. In this the non-linear variation of the degree of service or blocking probability with respect to the traffic intensity of the telecommunications system is observed. Thus, for ANN training and better adjustment of the hypothesis, it was necessary to start with constructing the data tables. The input data are the parameters of the number of channels and trunk traffic intensity measured in Erlangs. As output data, the

probability of blocking or GOS, varying for the latter, the percentage proportionality from zero to one for precision when the activation functions of each artificial neuron are applied.

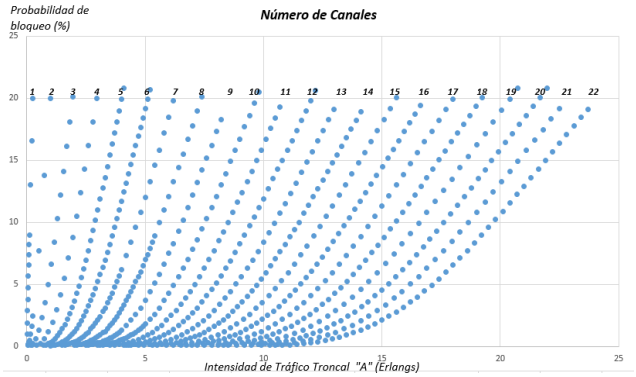


Figure-2. ErlangB chart up to 22 channels.

Likewise, the other data is tabulated until reaching 120 channels.

3. RESULTS

3.1 Multilayer ANN Design of Erlang B

The ANN model that best represents this type of function is the one made up of an input layer, two hidden layers and an output layer, as shown in Figure-3.

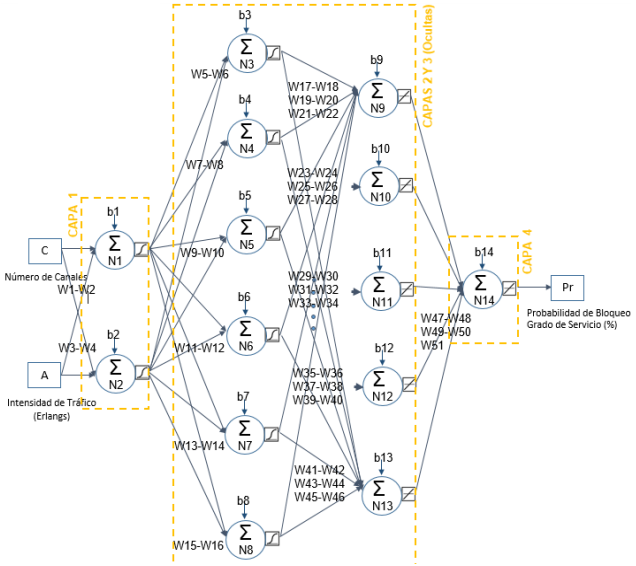


Figure-3. Erlang B function multilayer ANN.

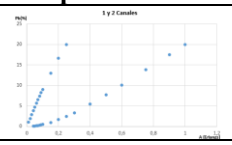
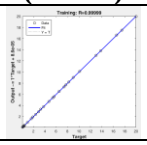
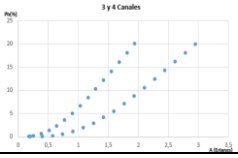
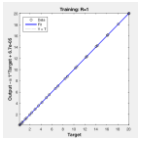
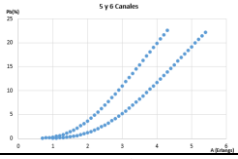
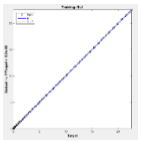
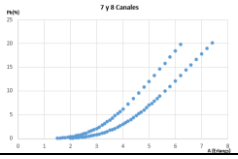
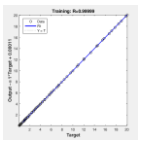
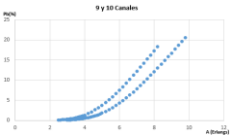
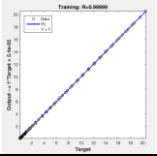
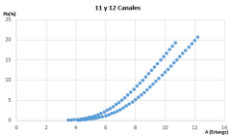
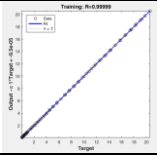
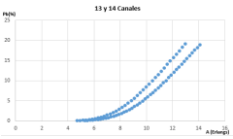
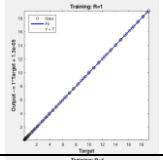
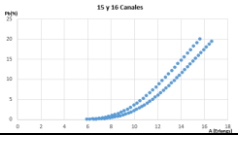
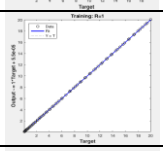
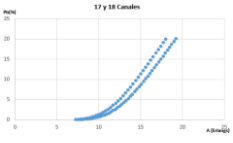
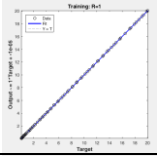
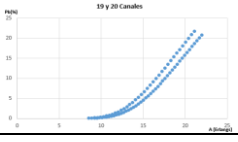
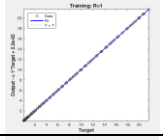
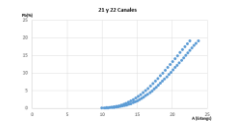
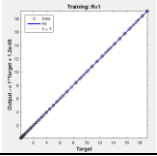
A more detailed description is presented in Table-2, where the number of neurons with their respective activation functions per layer is described.

Table-2. ANN configuration of Erlang B function.

Layer number	Number of neurons / Layer	Activation function / layer
1 (Input)	2	Logsig
2 (Hidden)	6	Logsig
3 (Hidden)	5	Purelin
4 (Output)	1	Purelin

By configuring the Levenberg-Marquardt learning algorithm, the best fits with regression functions are obtained by training the ANN with the input data (each pair of channel number and trunk traffic intensity) and the respective output (blocking probability), these characteristics, with some results, are detailed in Table-3:

Table-3. ANN training results for each pair of channels.

Erlang B chart scatter plot (Each pair of channels)	θ_j	**Muy** $J(\theta_0, \theta_1)$	**Regression (data fit)**
	0,0032	1e-05 0,000472	
	0,00382	1e-05 0,000238	
Erlang B chart scatter plot (Each pair of channels)	θ_j	Muy $J(\theta_0, \theta_1)$	Regression (data fit)
	0,015214	1e-06 0,000326	
	0,00413	1e-06 0,0006	
	0,00239	1e-06 0,000378	
	0,132	1e-06 0,000699	
	0,000152	1e-06 7,13e-05	
	0,0144	1e-06 7,59e-06	
	0,00422	1e-05 4,67e-05	
	0,00123	1e-06 1,66e-05	
	4,77e-08	1e-10 6,45e-06	

Each time the "Trainlm" algorithm is executed, the weights and oscillation units of the multilayer ANN are updated, serving these data as constants for the local variables of the different functions with a return that contain the characteristic equations of the network, being written in a non-vectorized way.

The ANN training process is practical when using the "Matlab" software tool since it contains the learning algorithms and performs the calculations in a vectorized way. With the parameters obtained from the best fit, it is possible to implement the forward propagation equations in the different object-oriented programming languages such as java and c ++.

In equations (3), (4), (5), and (6), each layer is mathematically modeled with its neural component, the respective activation functions, and the forward propagation from the input layer to the output layer:

The superscript indicates the layer, and the subscript indicates the number of the weight and the unit of oscillation (bias).

(**C**-Channels, **A**-Traffic Intensity of the trunk system "Erlangs", **Wn**-Weight, **bn**-bias, **Pr**-Probability of blocking or grade of service)

Layer 1 (2 neurons) Logsig activation function

$$n_{1,2}^{(1)} = g(b_{1,2} + \boldsymbol{C} * W_{1,3} + \boldsymbol{A} * W_{2,4}) \quad (3)$$

Layer 2 (6 neurons) Logsig activation function

$$n_{3,4,5,6,7,8}^{(2)} = g(b_{3,4,5,6,7,8} + n_1^{(1)} * W_{5,7,9,11,13,15} + n_2^{(1)} * W_{6,8,10,12,14,16} \quad (4)$$

Layer 3 (5 neurons) Purelin activation function

$$n_{9,10,11,12,13}^{(3)} = (b_{9,10,11,12,13} + n_3^{(2)} * W_{17,23,29,35,41} + n_4^{(2)} * W_{18,24,30,36,42} + n_5^{(2)} * W_{19,25.31,37,43} + n_6^{(2)} * W_{20,26,32,38,44} + n_7^{(2)} * W_{21,27,33,39,45} + n_8^{(2)} * W_{22,28,34,40,46}) \quad (5)$$

Layer 4 (1 neuron) Purelin activation function

$$\boldsymbol{Pr} = n_{14}^{(4)} = (b_{14} + n_9^{(3)} * W_{47} + n_{10}^{(3)} * W_{48} + n_{11}^{(3)} * W_{49} + +n_{12}^{(3)} * W_{50} + +n_{13}^{(3)} * W_{51}) \quad (6)$$

3.2 Multilayer ANN Training

The multilayer ANN training was done with the Matlab simulation tool. The functions established for the training of multilayer artificial neural networks, the layers, the number of neurons / layer, and the respective activation functions / neuron are configured.

In the case of interest, where the ANN of the non-linear function of ErlangB is modeled.

The best adjustment and obtaining of weights and biases is used a table for every 2 channels, with 110 input data and 55 output data.

The following pseudo-code shows the process for ANN training:

1: **Write** "Enter input data as channels and traffic intensity to training matriz"
2: **Read** input
3: **Write** "Enter output data as GOS to vector of training"
4: **Read** output
5: **Configure** ANN with layers, number neuron / layer and activation functions
6: **Configure** training parameters
7: **Call** the 'Levenberg marquart training function
8: **Validate** results and verify best training
End of Algorithm

Figure-4 shows the ANN architecture with the 4 layers and activation functions Logsig and Purelin for the non-linear function Erlang B.

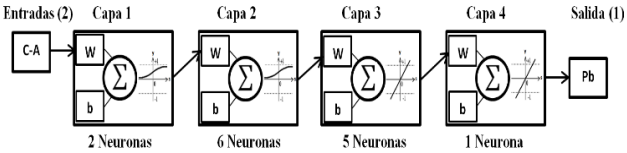


Figure-4. Multilayer ANN for Erlang B nonlinear function.

The descending gradient and the adaptive value mu are modified at each iteration until reaching the configuration of 1000 iterations with zero validation failures, and thus achieving the optimal fit (lower cost function) and better training with the weights and bias constants of the ANN:

The parameters of the hypothesis and the cost function are modified by applying the formulation of the descending gradient θ_j for ($j = 0$ y $j = 1$) where:

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \quad (7)$$

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \quad (8)$$

Where α is the learning index and θ_0, θ_1 are the parameters of the hypothesis.

At the beginning of the first iteration, the cost function presents a maximum value, then the descending gradient is also modified, until reaching the training objective with the best fit at 1000 epochs and the minimum mean square error of 0.000326 (for both channels 5 and 6).

The equation of this cost function $J(\theta_0, \theta_1)$ is:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) + y^{(i)})^2 \quad (9)$$

The hypothesis function $h_\theta(x)$ obtained is:

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x \quad (10)$$

Where the parameters θ_0, θ_1 vary depending on the data tabulation in correspondence to the number of channels used.

3.3 Model Native Application MVC

For the design of the native application in the Android Studio development environment, there are three parts that act dynamically being: the model, the view, and the controller. In the model, the reference framework and the starting point of the design are reflected, which for the project would be the function and the Erlang B multilayer artificial neural network system with all the mathematical components in a non-vectorized way and database management.

In the view, the interface is designed, and the basic objects of the algorithm are determined (text boxes, buttons, labels, and graphics) for the input, processing, and output of data; that must be present in an intuitive way to facilitate interaction between the person and the mobile device.

Regarding the controller, this refers to the software process that allows the functionality of the view designed in XML language, the Erlang B model, and the database. In this last part, the java classes, methods, and Android libraries are included.

Figure-5 shows the MVC of the native application:

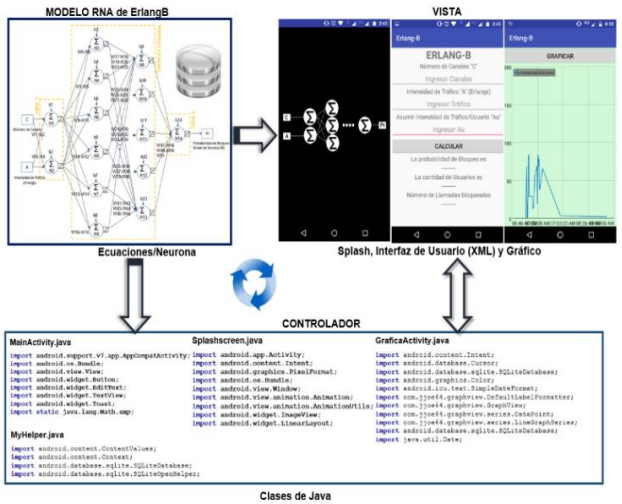


Figure-5. Android native MVC app model.

4. ANALYSIS OF RESULTS

4.1 Android App Functionality

When the application is opened, a first "splash" view is presented, which is an image with animation, indicating the multilayer ANN configuration with which the mobile application is designed. After six seconds, the second view is shown as an interface for interaction with the user; thus, the input and output data are displayed.

As input data for the Erlang B function, there are the variables "Channels", "Traffic intensity" of the trunk system, and "Traffic intensity" per user.

As output data, there are the variables "Probability of blocking or grade of service", "Number of users" and "Number of blocked calls".

The approach of the following example shows the results obtained in the executions of the app.

Example: Investigate the use of the Erlang B chart to obtain the number of users that a given system can support by providing a blocking probability or grade of service of 0.5%, in a non-queuing system with 5, 10, 20, and 100 channels.

Assume that each user generates 0.2 Erlang of traffic.

According to the blocking probability or degree of service of 0.5% that corresponds in the graph to 0.005 and the number of channels 5, 10, 20 and 100, the values of traffic an intensity are obtained.

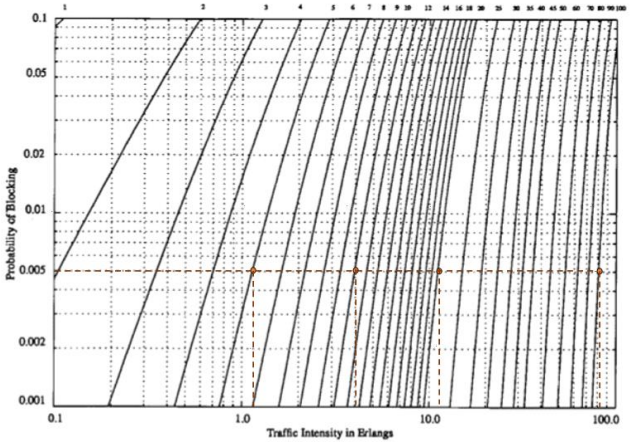


Figure-6. Erlang B chart.

Table-4. Traffic an intensity results.

Block Probability of 0.005 (0.5%)	
Channels (C)	**Traffic Intensity in Erlang (A)**
5	1.1
10	3.95
20	11.08
100	80.95

By assuming the traffic intensity of each user Au (Erlang) = 0.2. The next formula is used:

$$A = UAu \tag{11}$$

The number of users is calculated according to:

For C = 5 and A = 1.1

$$\frac{A}{Au} = U\ ;\ \frac{1.1}{0.2} = U;\ U = 5.5$$

For C = 10 and A = 3.95

$$\frac{3.95}{0.2} = U;\ U = 19.75 = 20$$

For C = 20 and A = 11.08

$$\frac{11.08}{0.2} = U;\ U = 55.4 = 56$$

For C = 100 and A = 80.95

$$\frac{80.95}{0.2} = U;\ U = 404.75 = 405$$

Table-5. Results of the number of users.

Block Probability of 0.005 (0.5%) and Traffic Intensity for each user Au (Erlang) = 0.2		
Channels (C)	**Traffic Intensity in Erlang (A)**	**Number of Users (U)**
5	1.1	5.5
10	3.95	19.75
20	11.08	55.4
100	80.95	404.75

In correspondence to the results shown in Table-5, it can be seen in Figure-7 how the number of users increases over time as the traffic intensity of the trunk system and the number of channels increases; having the same blocking probability or GOS of 0.005 (0.5%) and traffic intensity/terminal "Au" of 0.2 Erlangs.

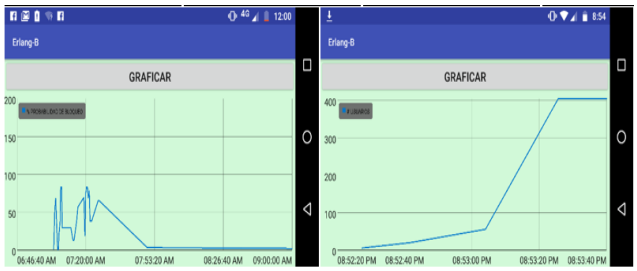


Figure-7. Graph of GOS and number of users vs. time.

In the applications found on the Internet, the HTML5 "Erlang B Calculator" and the Android "Erlang B / C" applications, the blocking probability can only be calculated with a precision of three and four decimal places, respectively; and without the possibility of entering the traffic intensity/user, being a necessary variable to obtain the number of users that the communication system can serve.

Below is the mobile app functionality test compared to Erlang B chart results.

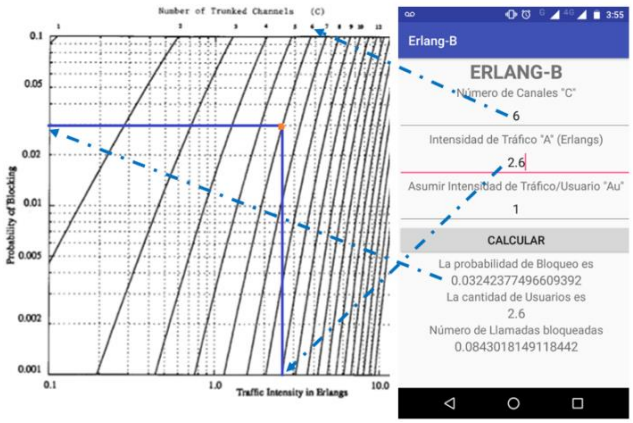


Figure-8. Comparison between Erlang B chart and App developer.

From the menu button, link "Graph", the data obtained from the blocking probability is sent to be inserted in the database and displayed in the two-dimensional graph, independent axis "time" and dependent axis "Probability of Blocking".

4.2 Data

The process of execution, training, and data validation in Matlab is carried out sequentially until reaching 120 channels.

For the case of the implementation of the native Android application, the values of weights and oscillation units, obtained from the best training sessions, were declared as global variables. Then, depending on the decision in entering the number of channels, they were called to the java functions so that the respective mathematical processes of the multilayer neural network are carried out.

5. DISCUSSIONS

In computational intelligence and telecommunications, this case study may be a reference for future research projects to be developed.

Although this proposed method requires a higher computational cost, it differs from other applications, because it allows obtaining precise data and optimal adjustments when applying the technique of neural networks and learning algorithms.

Other possible developments would be the implementation of the ANN model of the ErlangB function in FPGA and PSOC systems, Implementation of the artificial neural network model for the Erlang C function, Prediction models for the improvement of the quality of service according to real data obtained from telecommunications companies, ANN models for resource optimization (channels, time slots, radio channels, among others) and base station optimization model according to the statistical history of the number of calls made, the grade of service and intensity of trunk and user traffic.

CONCLUSIONS

The complexity of non-linear functions is related to the design and architecture of artificial neural networks; for the case study, it was possible to establish the multilayer ANN design of the Erlang B function with the variables of Channels, Traffic Intensity, and Probability of Blocking or GOS. With this methodology of Multilayer ANN design, training, algorithmic and mathematical treatment, it is possible to advance in the development of computational intelligence products that can be implemented in smartphones and embedded systems.

The development of the App for smartphones with Android operating system is useful as a simulation tool, since it differs from other applications due to the precision of the data, database management, graphing, flexibility, and applicability to computer systems communications.

REFERENCES

A. Harrel. 1988. Sharp bounds and simple approximations for the Erlang delay and loss formulas. Management Sciences. 34(8): 959-972.

Angus I. 2001. An introduction to Erlang B. and Erlang C. Telemanagement. 187: 6-8.

Eric Wong, Andrew Zalesky, Zvi Rosberg, and Moshe Zukerman. 2007. A new method for approximating blocking probability in overflow loss networks. Computer Networks. 51(11): 2958-2975.

Erlang B. and C. Calculator. GooglePlay. Available in: https://play.google.com/store/apps/details?id=dim.erlangbccalculator&hl=es

Erlang B. Calculator. Available in: http://www.erlang.com/calculator/erlb/

F. P. Kelly. 1986. Blocking Probabilities in Large Circuit-Switched Networks. Advances in Applied Probability. 18(2): 473-505.

Hagan Demuth, Beale *et al*. 1996. Neural network design. Pws Pub. Boston

Freeman R. L. 2015. Telecommunication system engineering. vol. 82, John Wiley & Sons.

L. Takacs. 1969. On Erlang's Formula. The Annals of Mathematical Statistics. 40(1): 71-78.

Pedro Ponce Cruz. 2011. Inteligencia Artificial, con aplicaciones a la ingeniería. Marcombo.

Rafael Lahoz Beltrá. 2004. Bioinformática simulación, vida artificial e inteligencia artificial. Ediciones Díaz de Santos, S.A, 28037 Madrid.

Robert Cooper, Daniel Heyman. 1998. Teletraffic theory and Engineering. Froehlich/Kent Encyclopedia of Telecommunications, Vol. 16, Dekker. 453-483.

Richard Parkinson. 2002. Traffic Engineering Techniques in Telecommunications. Infotel Systems Corporation.

Tibor Misuth, Ivan Baronak. 2011. Packet loss probability estimation using Erlang B model in modern VoIP networks. Published in: 7th International Conference on Electrical and Electronics Engineering (eleco). isbn: 978-1-4673-0160-2. 2: 259-263. 1-4 december, bursa, turkey.

Villy Iversen. 2001. Teletraffic engineering handbook.

William Lee. 1995. Mobile Cellular Telecommunications. Mc Graw Hill.