TCP performance estimation using neural networks modelling

B.V. Ghita[1], S.M. Furnell[1,2], B. Lines[1], and E.Ifeachor[1]

[1]Network Research Group, University of Plymouth, United Kingdom
[2]School of Computer and Information Science, Edith Cowan University, Perth, Australia
e-mail: info@network-research-group.org

Abstract

In recent years, the Internet has evolved from providing connectivity to hosts towards providing performance to applications, leading to significant interest in TCP performance modelling. Hitherto, mathematical approaches have been used to evaluate data transfers performance. Although such models were built upon a sound theoretical basis, they were validated largely using synthetic or endpoint controlled traffic and they are not suitable for short-lived transfers or clients with unknown behaviour, characteristics that are typical for the current Internet. The research presented in this paper aims to overcome these problems by using a supervised adaptive learning approach to build the relationship between TCP performance and the influencing parameters. The proposed model, using knowledge of past connections, was trained and tested using both synthetic and real network traffic. Comparison against the mathematical models showed that the proposed model provides more accurate estimates in the majority of the cases. The tests results indicated that the average error of the TCP throughput, estimated using the proposed model, was reduced to more than half the value obtained using the mathematical approach. The robustness and accuracy of the model allows its use in a variety of areas, such as network planning and evaluation of new TCP implementations.

Keywords

Neural network applications, Computer network performance, Modelling, Data flow analysis.

1. Introduction

Recent years have seen the Internet migrating from availability provisioning and rapid expansion towards slower growth and an enhanced interest towards quality provisioning. One of the areas that reflected directly this increase in performance-awareness was protocol performance modelling. Several studies, such as (Padhye et al 1998), and (Cardwell and Anderson 2000), focused on describing the evolution of connections time when using the Transmission Control Protocol (TCP), the underlying transport protocol for the World Wide Web. The aim of these studies was to mathematically replicate the protocol behaviour of TCP and to produce a performance estimate of the data transfer. The studies include a sound mathematical background which allows them to fully explain the evolution of the TCP connection from a theoretical perspective. The proposed models have been thoroughly tested on a variety of synthetic and controlled environments. However, these models also have several limitations: they are aimed at the congestion window evolution in time, in particular in the stationary stage, rather than performance; they are suited to long data connections, not likely to occur in real traffic conditions; they rely on knowledge of the TCP behaviour of the endpoints, using as a starting point a fully defined TCP implementation. Studies have shown that typical implementations differ from the existing standards (Floyd and Padhye 2001). This

paper proposes a model that aims to overcome the lack of robustness in current mathematical approaches but also to produce a higher accuracy when predicting performance.

Neural-network-based approaches were extensively used in the networking area as an alternative to existing statistical or mathematical solutions. A few examples of successful applications (Catania et al 1996), (Cheng and Chang 1996), and (Ramaswamy and Gburzynski 1999) proposed neural networks as an alternative to improve quality of service control in ATM networks. The studies have shown that a knowledge-based approach may lead to better results when faced with the variable nature of traffic, in comparison with statistical approaches.

After highlighting the limitations of current mathematical models, the second section will then introduce a knowledge-based approach that aims to provide better results when modeling TCP performance. Section two will present the IDA architecture, adapted for this task, and will expand on each of the actual stages. Section three will describe the preliminary stages of the validation, and then section four will describe the actual validation tests. Section five concludes the paper.

2. Neural-based performance model

2.1. Introduction

This section proposes a novel prediction model, based on intelligent data analysis (IDA) techniques. The approach used aims to bridge the relationship between network and transfer conditions on one side and the resulting performance on the other. The IDA was considered appropriate due to the features that range beyond mathematical models, such as unknown client behaviour and uncertain network events. It must be stressed that the primary aim of the proposed model is not to compete with the existing ones, but to cover a different area of TCP performance analysis by using a different approach, which may be summarized as: a*nalyse the performance of a comprehensive range of current traffic from real networks and, based on this knowledge, predict the performance of TCP traffic for any combination of network conditions within the known scope*. This is, in fact, the fundamental distinction between mathematical models and for the performance prediction model proposed in this project: use prior knowledge to predict overall performance instead of trying to determine the evolution of the TCP connection. The proposed solution does not provide any information about the evolution of a connection in time, but only about the overall performance (in terms of throughput) that is likely to achieve, based on the network and endpoint conditions.

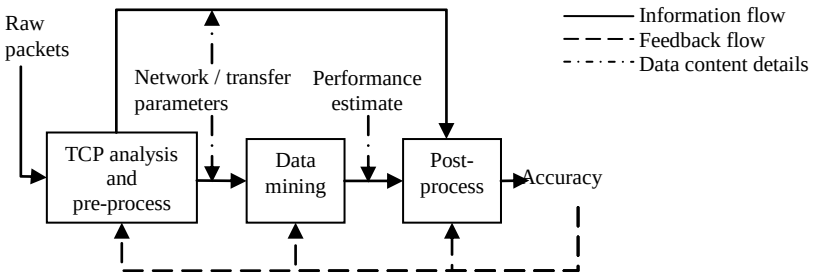

Figure 1 - - IDA processing diagram – basic representation

The generic IDA model described in (Fayyad *et al* 1996) was adapted for the needs of this study, resulting in the schematic diagram from Figure 1. The meaning of the processing blocks from the diagram will be detailed in the reminder of this section.

The input to the process is the network traffic – *Raw packets*, representing the raw data captured from the packet headers. The capturing can be done for either live processing (where data is fed to the IDA right away) or offline processing – the content of the packets which is stored in a file and analyzed at a later time by the IDA. The raw header content is unusable for data mining. The next step is to acquire knowledge and transform data in a meaningful format, using TCP analysis. For the scope of this project, the method applied was the one described in (Ghita *et al* 2001). The next step, *Pre-processing*¸ is essential for the performance of the entire algorithm: the parameters obtained from the raw input during the previous step must be put in a form suitable for the data-mining algorithm. It is at this stage where data may be transformed, filtered, scaled in order to fit the chosen algorithm.

The *Data mining* part is, aside from highlighting the importance of data pre- and post processing, the core of the IDA process. The aim of this stage is to determine a relationship between different instances of the input parameters and the variable that is predicted. It uses Network and transfer parameters during the training phase to produce a function to map the TCP behaviour onto resulting performance - the duration of the connection. In the testing phase the data mining engine is presented with unseen samples. The set of rules and functions established in the training phase is then used to provide at the output an estimated value, the Performance Estimate. Connection Parameters represents an instance of the set of variables that define a connection (network conditions, endpoint types, and file size). They are extracted using the proposed monitoring method from a network trace, in the same way as it happens during the training phase. The output of the data mining algorithm may require further processing in order to interpret its significance. It is the combined task of the data mining output and the *Post processing* block to perform this further analysis in order to evaluate the success of the method.

3. Applying IDA to TCP performance modelling

3.1. Data collection and pre-processing

The first issue is what data should be fed into the data mining algorithm. From the studies that developed mathematical TCP models and the overall behaviour of TCP clients, it became apparent that network parameters, as seen by the endpoint, have a vital impact on the performance of the transfer. In addition, the connection parameters (initial and maximum congestion window size and file size) were considered. The output data from the TCP analysis was filtered to remove the outliers of the inputs and output distributions. The filter removed both the extreme values for throughput *and* the input variables (delay, loss, congestion window): eliminate the extreme 1% from the input variables domains and the extreme 3% of the output variables domains. These successive filters were applied simultaneously – the filter for each variable was applied to the original dataset, not on the remaining data. All these measures aimed to balance between excessive removal of data and reduction of the prediction errors.

3.2. Data analysis – procedure and algorithms

The data analysis step used the Stuttgart Neural Network Simulator (SNNS) (SNNS 2004) a generic neural network engine that includes a comprehensive list of neural network algorithms to use for specific prediction / classification tasks. Two separate neural networks were applied, depending on whether the connections encountered any packet loss or not. As a result, two distinct sets of inputs were used: a smaller one (three parameters), with no information about loss, used for no-loss connections, and a slightly larger one (five parameters), which included loss information. In spite of this difference, the main characteristics of the two data sets remained largely the same: small number of inputs (either 3 or 5, depending whether loss was modelled or not) and large number of samples (up to thousands of samples).

3.3. Interpretation of the results and implementation

Three indicators were used to determine the accuracy of the SNNS prediction. First alternative was graphical display of results - very good but human-observation based, therefore unusable for analysis. This involved plotting each of the n real values s_i versus the predicted values p_i: $f(s_i) = p_i, i = \overline{1,n}$. In the ideal case, 100% accurate prediction, the result is the identity function, f(x)=x. Although convenient, the method cannot be used for analytical purposes, which is why average relative error was employed as a second alternative, used to statistically compare the predicted and the real values. However, relative error is less relevant if the variable has a low standard deviation, therefore correlation factor, was used to eliminate poor prediction undetectable through relative error for such variables.

Using raw tcpdump traces as input, the implementation followed the IDA processing stages:

- Pre-processing - scripts and programs to produce a database of TCP connection samples. The network trace analysis was performed using the method from (Ghita *et al* 2001).
- Processing – scripts to communicate with *batchman*, the SNNS environment: train/test the neural network with the training /testing datasets, provide mechanisms for early stopping
- Post-processing – scripts to evaluate the efficiency of the prediction, based on the plot of predicted versus real values, the Mean Square Error (MSE), and the correlation factor

4. Preliminaries

4.1. Sources of data

The training/testing data came from three main sources:

- ***NS*** simulations - Berkeley Network Simulator (ns) (NS2 2004) suite was chosen from the plethora of network simulator choices. The ***NS*** traces were obtained by randomly generating TCP connections over network infrastructures.
- Semi-controlled traffic - produced using an automatic retrieval tool. The data collection was based on the idea used by Neil Cardwell to prove the efficiency of his TCP performance model in (Cardwell and Anderson 2000), using the *Random Yahoo Link* (RYL) - a Common Gateway Interface (CGI) script within the Yahoo website that

redirects the HTTP client to a random web page (RYL 2004). The retrieval used *wget* (Wget 2004) as the HTTP client, and *tcpdump* for packet capturing

- Uncontrolled traffic - real network traffic traces. The capturing was performed at the University of Plymouth (UoP) network, using *tcpdump*, with no artificial / controlled element involved; all the connections were due to genuine (human) requests from the endpoints.

4.2. Data analysis

The transfer during connections without losses is affected only by the congestion window characteristics (initial value), the amount of data to transfer, and the round trip delay between the two endpoints. As a result, the model for these connections used only three inputs: the amount of data to transfer, the initial value of the congestion window (both in bytes), and the estimated round trip delay (in milliseconds). Initially, the maximum value of congestion window was added to the list, to consider the case when the congestion window increase is limited by the receiver advertised window. However, the connection analysis revealed that this did not happen throughout any of the datasets, due to the value of the advertised window itself. Some of the receivers from the UoP dataset had implemented, however, a window scale mechanism (Funashi 1989) to avoid such limitations. To make the neural network model comparable with the mathematical one and, more importantly, to allow its usage on generic loss rates, the list of loss-related parameters included only two variables: the total loss rate and the estimated timeout. The total loss rate was a sum of the visible and inferred fast retransmissions and inferred timeouts. The estimated timeout period was replaced with the total duration of timeouts. The resulting list of inputs for connections with losses was: the amount of data to transfer, the size of the initial congestion window, the round trip delay, the estimated retransmission timeout, and the loss rate. The estimation results from the neural network models were compared with the current existing mathematical models in order to asses whether they are superior in terms of accuracy and robustness. The comparison was made in terms of relative error, both in the average value and actual distribution, as well as using the correlation factor from the two methods.

4.3. Preliminary tests – Connections without losses

The purpose of the preliminary analysis was to evaluate which neural network (in terms of structure, algorithms, and parameters) is likely to lead to a greater accuracy for a set of connections that do not encounter any loss events.

There were three main parameters to vary when applying a neural network to a dataset: the structure of the network, the method applied, and the amount of data to be fed to the network. The structure of the network included variables relating to the number of hidden layers, the connectivity between neurons, and the number of neurons for each hidden layer. The method applied was the most complex part, with several sub-divisions: establishment of the training algorithm together with the initialization and update functions, identification of the optimum parameters for the chosen network, followed by detection of the early stopping decision that would lead to highest accuracy. Due to the size limitations, this paper does not include the procedure used to select the optimum neural network structure, parameters, and early stopping decision, but only presents briefly the results of these preliminary tests.

The design of the networks started by considering the neural network training algorithm used, the number of input and output neurons, and then produced several alternatives of hidden layer(s). The training algorithm used was back propagation with momentum and flat spot elimination[1], based on recommendations from (Haykin 1999). In which regards the inputs and outputs of the neural network, for the loss-free case, the dataset included samples with three inputs: the connection size and initial congestion window (both measured in bytes), and the round trip delay, and a single output – the duration of the data transfer. Based on these inputs/outputs, a fully connected neural network was generated, with the structure 3-6-3-1 (two hidden layers, first / second layer with six / three neurons). Similarly, for the neural network aiming to model the connection with losses, the preferred solution was to use a structure with two hidden layers. The network was fully connected, with 5 inputs (the three from the no-loss network plus the estimated retransmission timeout, and the loss rate), and a structure of 5-10-5-1. The early stopping criteria were decided based on the characteristics of the datasets, using prior work in the area (Amari *et al* 1997). After running the evaluation batches, the minimum MSE for connections with losses was obtained for (η,τ,μ,c)=(0.07,0.0,0.5,0.1), while the minimum NSE for connections with losses was obtained for (η,τ,μ,c)=(0.03,0.0,0.3,0.1). These combinations of variables were used for the loss subset validation section.

5. Validation tests – Connections without losses

5.1. The NS dataset

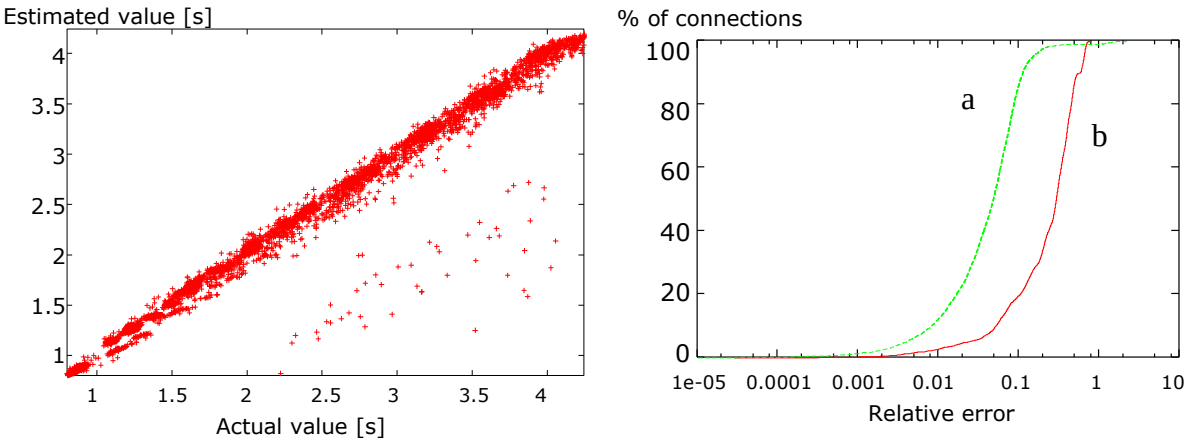

Figure 2 – Left - Plot of the real values vs. estimated values, as resulting from the *NS* dataset. Right - Cumulative distribution of the relative error for the RYL dataset using the (a) neural network model and (b) mathematical model

The dataset was applied to a (2-6-3-1) neural network and trained exhaustively using 10000 cycles, employing the set of parameters (η,τ,μ,c)=(0.03,0.0,0.3,0.1). At the end of the training session, the minimum registered MSE was 0.00222, resulting after 4720 cycles. The neural network followed the real values accurately, as exhibited by the graph in Figure 2 - left.

[1] The chosen variant of back-propagation has two improvements when compared with its predecessors: the momentum term μ that absorbs any eventual oscillations of the resulting error and the flat spot elimination term c, which allows the network to surpass possible flat spots (local minima) on the error surface. Aside from these two terms, the algorithm also includes the typical learning rate η and tolerance threshold τ, which define back propagation.

The next step was to apply the ***NS*** dataset to Cardwell's model (Cardwell and Anderson 2000) in order to determine how accurate the mathematical model is when compared with the neural network model. The results produced by this second model were then compared with the results from the neural network model. The statistical results of the comparison are presented in Table 1. It may be observed that, on average, the neural network model outperforms the mathematical model in terms of accuracy. The table also includes a column for the correlation between the predicted values and the real values in order to illustrate that the accuracy of the neural network was not due to the narrow spectrum of the output variable, but to learning the variations in transfer duration produced by changes in the amount of data transferred and the RTT values.

Model	Average relative error	Stdev. of relative error	Correlation
Mathematical	0.29268	0.18421	0.96907
Neural	0.03052	0.04785	0.98701

Table 1 - Comparison of the resulting average figures for the *NS* dataset, using the mathematical and the neural network models

Table 1 shows that the neural network models perform on average radically better than the mathematical model. A second graph was produced to illustrate how the neural network and mathematical models perform throughout the dataset. A separate graph, shown in Figure 2 - right, was produced to compare the relative error produced by the two models. The plot confirms the average figures, indicating that the neural network model outperformed the mathematical model throughout the vast majority of the dataset

The results obtained from the ***NS*** dataset indicated that neural networks may provide better performance estimate, compared to mathematical models, of TCP transfers that encountered no losses. However, the other main objective of the method is to provide a robust estimator. This is why the next step was to apply and test the neural network on a more realistic environment, i.e. the dataset produced from the RYL connections.

5.2. The RYL dataset

The training process was similar and used the same dataset as the one from the preliminary tests. The network was trained for 10000 cycles and it reached a minimum of 0.00658 for MSE after 9230 cycles. It is visible from scatter shown in Figure 3 – left that the accuracy of the neural network decreased. However, the neural network still performed better than the mathematical model, as shown in Table 2 and also when plotting the relative error distributions of the two models, as shown in Figure 3 - right.

Model	Average relative error	Stdev. of relative error	Correlation
Mathematical	0.58665	0.45804	0.83474
Neural	0.17993	0.21464	0.89938

Table 2 - Comparison of the resulting average figures for the RYL dataset, using the mathematical and the neural network model

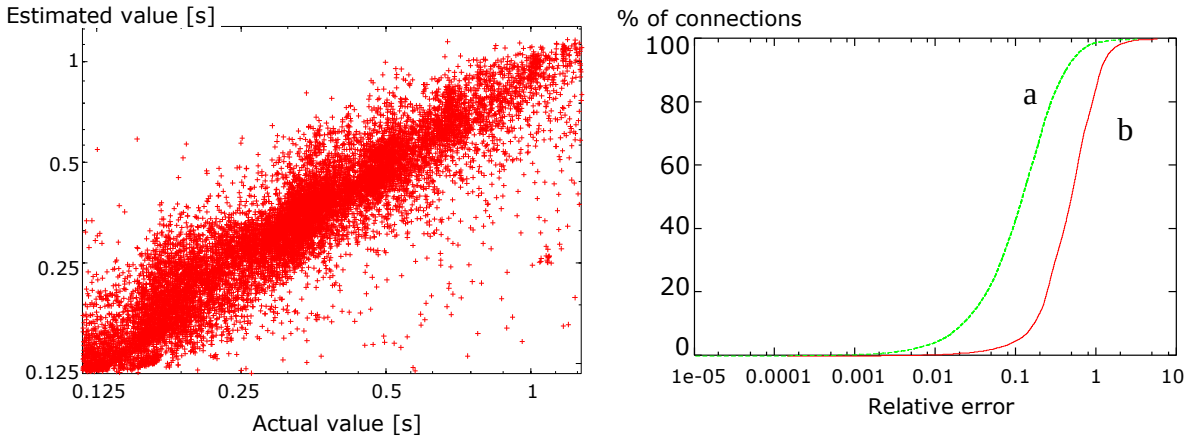

Figure 3 - Left - Plot of the real values vs. estimated values, as resulting from the RYL dataset. Right - Error distribution for the RYL dataset using the (a) neural network model and (b) mathematical model

5.3. Generalization – the UoP dataset

This dataset was expected to provide the full generalisation for the efficiency of the method. With the ***NS*** dataset, the senders and receivers all had the same behavior (not influenced by the attempts to vary it); with the RYL dataset, the senders varied and used unknown implementations, but the receiver used the same TCP implementation throughout the experiments (the TCP/IP implementation from SuSE Linux). The UoP data resulted from connections with unknown entities at both ends, using most likely a variety of implementations. The dataset was generated from a 1-hour trace and had 21629 samples, reduced to 18545 after filtering. The dataset was trained with the 3-6-3-1 network for 10000 cycles, reaching a minimum MSE of 0.01331 after 1770 cycles.

Even for this generalization case, in spite of the slight overtraining that may be noticed (the visible horizontal line at the estimated value of 1.3 seconds from Figure 4 - left), the neural network appears to perform better than the mathematical model. A comparison of the average figures may be seen in Figure 4 – right.

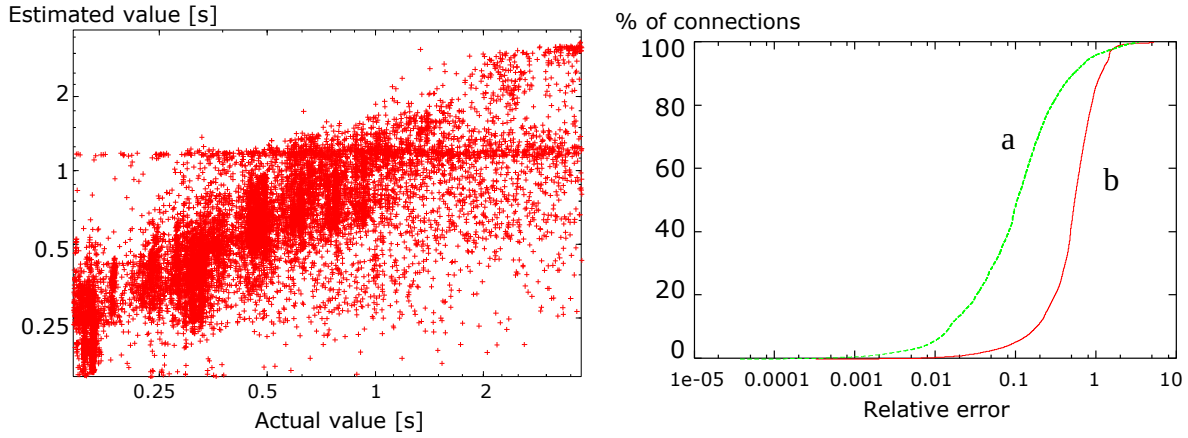

Figure 4 - Left - Plot of the real values vs. estimated values, as resulting from the UoP dataset. Right - Error distribution for the UoP dataset using the (a) neural network model and (b) mathematical model

Model	Average relative error	Stdev. of relative error	Correlation
Mathematical	0.60807	0.40234	0.62782
Neural	0.37268	0.41053	0.70027

Table 3 - Comparison of the resulting average figures for the UoP dataset, using the mathematical and the neural network model

It may be observed from the above table that the difference between the mathematical and neural models is smaller in comparison with the previous datasets. Nevertheless, the average error produced by the mathematical model is approximately 60% higher than the one resulting from the neural model. The comparison of the relative errors from the neural network and the mathematical model is presented in Figure 4 - right. The figure confirms the average results, showing higher accuracy for the neural model than the results of the mathematical model.

6. Validation tests –connections with losses

6.1. NS traces

The ***NS*** loss subset had 3396 samples (after filtering). The neural network was trained with the 4-4-2 neural network, using the combination of parameters (η,τ,μ,c)=(0.07,0.0,0.5,0.1). The network converged in 4160 cycles, to an MSE of 0.02887. Figure 5 – left shows a plot of the real values vs. the neural network results.

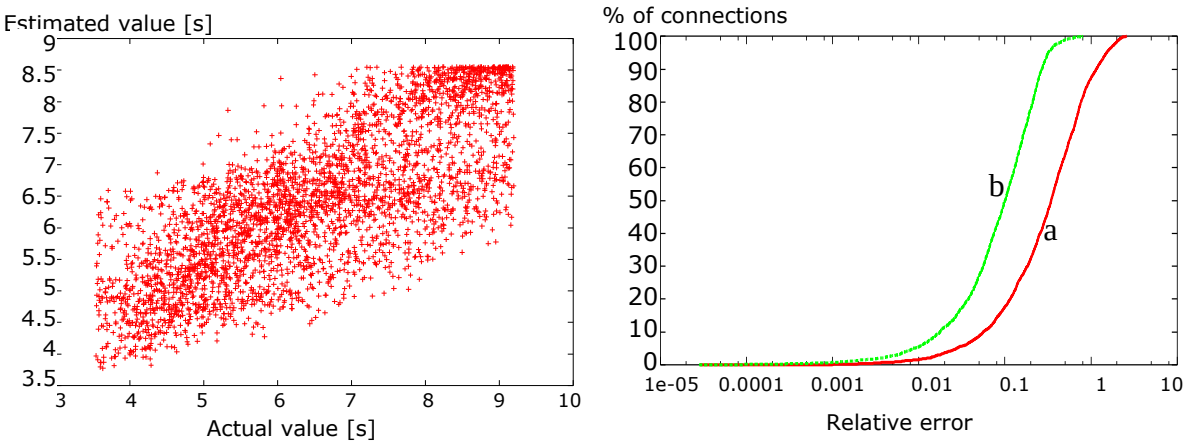

Figure 5 – Left - Plot of the real values vs. the neural network estimated values, from the *NS* loss subset. Right - Cumulative distributions of the relative error resulting from using the (a) mathematical model and (b) neural network model with the *NS* loss subset

The subset was applied in parallel to an implementation of the mathematical model in order to compare the accuracy of the two solutions. The resulting cumulative distributions of the relative errors, are shown in Figure 5 – right. The neural network outperformed the mathematical model throughout the dataset, as shown by the average values in Table 4.

Model	Average relative error	Stdev. of relative error	Correlation
Mathematical	0.48882	0.46993	0.42984
Neural	0.12695	0.10860	0.76117

Table 4 - Comparison of the resulting average figures for the *NS* loss subset, using the mathematical and the neural network model

6.2. UoP traces

The validation test consisted of training the UoP subset containing connections that encountered losses (5991 samples) with the 5-10-5-1 neural network. Figure 6 – left shows a comparison of the real values versus the output of the neural network model. It may be seen that the plot from Figure 6 – left does not illustrate a very good mapping between the actual and estimated values. The mapping also seems to be incomplete from a domain perspective: a large proportion of the actual values are situated in the [0.1s;1.0s] domain, while the estimated value seems to have a lower-bound around the value of 1s. As expected, this improper mapping leads to worse results when calculating the relative error of the prediction method and comparing it with the mathematical model.

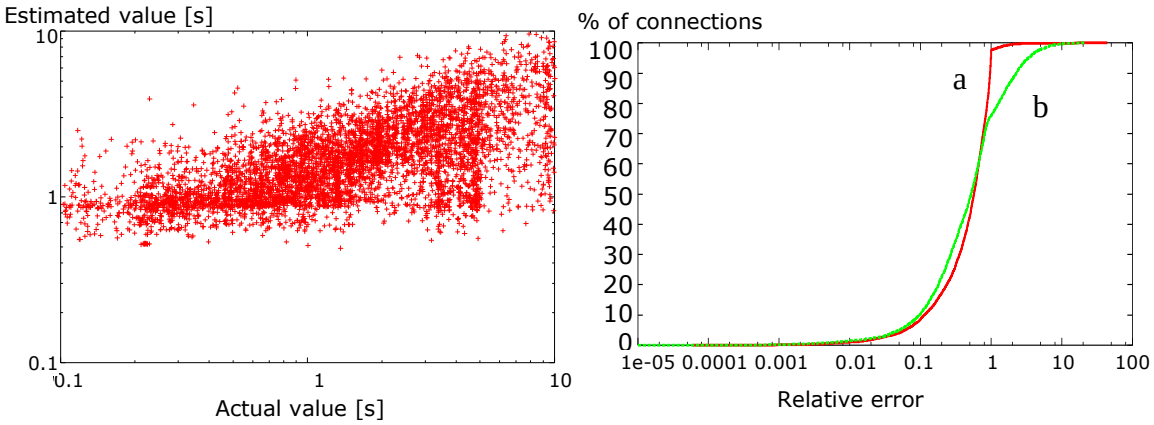

Figure 6 - Left - Plot of the real values vs. the neural network estimated values, as resulting from the UoP loss subset. Right - Accuracy of the (b) NN model and (a) mathematical model for the UoP loss subset

The accuracy of the two models is presented in Figure 6 – right. The figure shows that the neural network manages to outperform the mathematical model for more than half of the connections. The difference appears in the average error values too, shown in Table 5.

Model	Average relative error	Stdev. of relative error	Correlation
Mathematical	0.584	0.74837	0.46721
Neural	0.91511	1.34159	0.604

Table 5 - Comparison of the resulting average figures for the UoP loss subset, using the mathematical and the neural network model

7. Conclusions

This paper proposed a neural network-based model to predict the performance of TCP connections based on the network characteristics and endpoint variables. The validation studied the accuracy of the proposed neural network model when applied on traces consisting of either purely synthetic traffic or from real traffic. Two types of data, including connections with or without losses, were studied in separate rounds of tests. The validation tests consisted of using the resulting parameters to train the respective dataset, then to test its accuracy. The results were then observed and compared with the output of a mathematical model implementation. The comparison looked at the overall relative error, the distribution of the relative error, and the correlation factor between the actual and the predicted values.

The validation tests have shown that the proposed model provides an overall better accuracy when compared against the mathematical model using the three factors mentioned above. The tests performed on the no-loss subsets indicated a nearly ten-fold improvement of the average relative error in the case of ***NS***-generated data between the mathematical and the neural-based model. The improvement was also high in the case of RYL traces, where the relative error obtained with the mathematical model decreased by approximately 70% when applying the neural model, and by 40% in the case of real traffic collected from the UoP backbone. The results obtained from the trace subsets that included connections with losses were two-folded: the ***NS***-generated trace led to a 75% reduction of the overall relative error between the two models. Unfortunately, the UoP loss subset led to better prediction results when applying the mathematical model. However, after analyzing the plots of the values and of the corresponding relative errors, it was concluded that the poor accuracy may have been caused by neural network limitations. As an overall conclusion from the performed validation tests, the proposed neural-based model provides a better alternative to mathematical models in terms of accuracy. This accuracy improvement, together with the robustness of the method, provides a better described relationship between application performance and the conditions that influence it. The improvement also opens new avenues in the areas that relate to performance provisioning, such as: network planning, network control, and TCP implementations testing.

8. References

Amari S., Murata N., Muller K., Finke M., Yang H., 1997, "Asymptotic Statistical Theory of Overtraining and Cross-Validation", *IEEE Transactions on Neural Networks*, Vol. 8, No. 5, pp. 985-996

Cardwell N., Savage S., Anderson T., 2000, "Modelling TCP Latency", *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March

Catania V., Ficili G., Pallazo S., Panno D., 1006, "A Comparative Analysis of Fuzzy Versus Conventional Policing Mechanisms for ATM Networks", *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, June

Cheng R.G., Chang C.J., 1996, "Design of a Fuzzy Traffic Controller for ATM Networks", *IEEE Transactions on Networking*, vol. 4, no.3, June

Fayyad U., Shapiro G.P., Smyth P., 1996, "*From Data Mining to Knowledge Discovery in Databases*", AI Magazine, Fall

Floyd S., Padhye J., 2001, "On inferring TCP behavior", *Proceedings of the ACM SIGCOMM 2001*, August 27-31, 2001, San Diego, CA, USA. ACM,

Ghita B., Furnell S.M., Lines B., Ifeachor E., 2001, Non-intrusive IP Network Performance Monitoring for TCP Flows, *Proceedings of IEEE ICT2001*, pp290-295, 4-7 June, Bucharest, Romania

Ghita B., Furnell S.M., Lines B., Ifeachor E., 2002, "Endpoint study of Internet paths and web pages transfers", *Proceedings of the Third International Network Conference (INC 2002)*, Plymouth, UK, 16-18 July 2002, pp261-270

Haykin S., 1999, "Neural networks: a comprehensive foundation" 2nd edition, Prentice Hall

Jacobson V., 2004, *tcpdump source code*, ftp://ftp.ee.lbl.gov/tcpdump.tar.Z

Kennington, 2004, "*Simulation software links*", http://www.topology.org/soft/sim.html

NS2, 2004, "*The Network Simulator – ns2 homepage*", http://www.isi.edu/nsnam/ns/

Padhye J., Firoiu V., Towsley D., Kurose J., 1998, "Modelling TCP Throughput: A Simple Model and its Empirical Validation", *Proceedings of SIGCOMM '98*, Vancouver, CA

Ramaswamy S., Gburzynski P., 1999, "*A Neural Network Approach to Effective Bandwidth Characterization in {ATM} Networks*", in Performance Analysis of ATM Networks, IFIP vol. 4, Demetres Kouvatsos, editor, Kluwer Academic Publishers

RYL, 2004, "Random Yahoo Link page", http://random.yahoo.com/bin/ryl

SNNS, 2004, "Stuttgart Neural Network Simulator", http://www-ra.informatik.uni-tuebingen.de/SNNS/

WGET, 2004, "GNU wget homepage", http://wget.sunsite.dk/