Evaluation of Network Traffic Prediction Based on Neural Networks with Multi-task Learning and Multiresolution Decomposition

Melinda Barabas, Georgeta Boanea, Andrei B. Rus, Virgil Dobrota
Technical University of Cluj-Napoca
26–28 George Baritiu Street, 400027 Cluj-Napoca, Romania
Email: {Melinda.Barabas, Virgil.Dobrota}@com.utcluj.ro

Jordi Domingo-Pascual
Universitat Politècnica de Catalunya (UPC)
1–3 Jordi Girona Street, 08034 Barcelona, Spain
Email: Jordi.Domingo@ac.upc.edu

***Abstract*—Network traffic exhibits strong correlations which make it suitable for prediction. Real-time forecasting of network traffic load accurately and in a computationally efficient manner is the key element of proactive network management and congestion control. This paper compares predictions produced by different types of neural networks (NN) with forecasts from statistical time series models (ARMA, ARAR, HW). The novelty of our approach is to predict aggregated Ethernet traffic with NNs employing multiresolution learning (MRL) which is based on wavelet decomposition. In addition, we introduce a new NN training paradigm, namely the combination of multi-task learning with MRL. The experimental results show that nonlinear prediction based on NNs is better suited for traffic prediction purposes than linear forecasting models. Moreover, MRL helps to exploit the correlation structures at lower resolutions of the traffic trace and improves the generalization capability of NNs.**

***Keywords*—multiresolution learning, multi-task learning, neural networks, prediction**

I. Introduction

The main purpose of forecasting is to use historical data in order to predict the behavior of a system, by modeling it as a black-box [1]. Traffic prediction plays an important role in guaranteeing Quality of Service (QoS) in IP networks due to the diversity of services and because of the increased volume of real-time network applications. Forecasting algorithms can be embedded into network management systems to improve the global performance of the network and to achieve a balanced utilization of the resources. Traffic prediction can be useful for dynamic routing, congestion control and prevention, autonomous traffic engineering, proactive management of the network, etc.

Upon the occurrence of congestion in the network, a traditional routing protocol cannot react immediately, resulting in packet loss, additional delay and jitter, as well as services with severely degraded quality [2]. Prediction can be used by a network device in the self-adaptation process for optimizing its own performance. Thus, proactive decision-making is possible based on the predicted evolution of traffic on certain links, as opposed to reacting to past events. Thanks to the early warning, a prediction-based approach will be faster, in terms of congestion identification and elimination, than reactive methods which detect congestion through measurements, only after it has significantly influenced the network operation.

The prediction of network traffic parameters is possible because they present a strong correlation between chronologically ordered values. Their predictability is mainly determined by their statistical characteristics. According to [3], network traffic is characterized by: self-similarity, multiscalarity, long-range dependence (LRD) and a highly nonlinear nature.

Several methods have been proposed in the literature for network traffic forecasting. These can be classified into two categories: linear prediction and nonlinear prediction. Choosing a specific forecasting technique is based on a compromise between the complexity of the solution, characteristics of the data and the desired prediction accuracy. The most widely used traditional linear prediction methods are: *a)* the ARMA/ARIMA model [1], [4], [5], [6], [7] and *b)* the Holt–Winters algorithm [1], etc. The most common nonlinear forecasting methods involve neural networks (NN) [1], [3], [4], [8]. NNs can be combined with: *a)* multi-task learning [9], [10] or *b)* multiresolution learning [2], [11], [12], [13], etc. Although some articles state that linear prediction models are unable to describe the characteristics of network traffic [4], other studies confirm the practical usability of linear predictors for real-time traffic prediction [7]. Thus, it remains unclear which predictors provide the best performance, being in the same time simple, adaptable and accurate.

In this paper we consider the problem of forecasting the transfer rate, i.e. given a set of transfer rates observed on a specific link, we try to predict its future values. We chose the prediction of this parameter because this is the basic QoS parameter, i.e. if the demands regarding the transfer rate are not met, the other QoS parameters (delay, jitter, packet drops) will be affected seriously. In the following, we demonstrated that the prediction of future network traffic load based on recent observations is possible, with a certain accuracy, in a computationally efficient manner.

The rest of this paper is organized as follows. Section II gives a brief introduction to traditional forecasting techniques. In Section III neural network traffic predictors with multi-task training and multiresolution learning approaches are described.

Section IV lists the performance metrics used for prediction accuracy evaluation. The experimental results are presented in Section V as a comparative study with various types of predictors applied to real-world network traffic traces. Finally, Section VI concludes the paper and discusses future work.

II. Traditional Time Series Forecasting Models

In this section we give a brief introduction to various predictors based on traditional statistical techniques, such as ARMA(Autoregressive Moving Average), ARAR (Autoregressive Autoregressive) and HW (Holt–Winters) algorithm.

A. ARMA model

The family of ARMA processes is one of the most popular statistical methods used for modeling and forecasting linear time series. ARMA models rely on a linear combination of autoregressive (AR) and moving average (MA) components.

The time series $\{X_t\}$ is called an ARMA(p,q) process if $\{X_t\}$ is stationary (i.e. its statistical properties do not change over time) and :

$$X_t - \phi_1 X_{t-1} - \ldots - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q} \quad (1)$$

where $\{Z_t\} \approx \text{WN}(0,\sigma^2)$ is white noise with zero mean and variance σ^2 and the polynomials $\phi(z) = 1 - \phi_1 z - \ldots - \phi_p z^p$ and $\theta(z) = 1 + \theta_1 z + \ldots + \theta_q z^q$ have no common factors [14].

The identification of a zero-mean ARMA model which describes a specific dataset involves the following steps [14]: *a)* order selection (p,q); *b)* estimation of the mean value of the series in order to subtract it from the data; *c)* determination of the coefficients $\{\phi_i, i = \overline{1,p}\}$ and $\{\theta_i, i = \overline{1,q}\}$; *d)* estimation of the noise variance σ^2. After the validation of the model, predictions can be made recursively using:

$$\hat{X}_{n+1} = \begin{cases} \sum_{j=1}^{n} \theta_{nj}(X_{n+1-j} - \hat{X}_{n+1-j}), \text{if } 1 \le n \le m \\ \sum_{j=1}^{q} \theta_{nj}(X_{n+1-j} - \hat{X}_{n+1-j}) + \phi_1 X_n + \ldots + \\ +\phi_p X_{n+1-p}, \text{if } n \ge m \end{cases} \quad (2)$$

where $m = max(p,q)$ and θ_{nj} is determined using the innovations algorithm.

To fit a model to a nonstationary time series we use ARIMA (Autoregressive Integrated Moving Average). Fitting an ARIMA model to the original nonstationary dataset is equivalent with determining the ARMA model for the differentiated dataset. The ARIMA(p,q,d) process is described by:

$$\phi(B)(1-B)^d X_t = \theta(B) Z_t \ , \quad (3)$$

where ϕ and θ are polynomials of degree p and q respectively, $\nabla = 1 - B$ represents the differencing operator, d indicates the level of differencing and B is the backward-shift operator, i.e. $B^j X_t = X_{t-j}$ [4].

B. ARAR algorithm

The ARAR algorithm applies memory-shortening transformations, followed by modeling the dataset as an AR(p) process: $X_t = \phi_1 X_{t-1} + \ldots + \phi_p X_{t-p} + Z_t$.

The time series $\{Y_t\}$ of long-memory or moderately long-memory is processed until the transformed series can be declared to be short-memory and stationary:

$$S_t = \psi(B) Y_t = Y_t + \psi_1 Y_{t-1} + \ldots + \psi_k Y_{t-k} \ . \quad (4)$$

The autoregressive model fitted to the mean-corrected series $X_t = S_t - \bar{S}, t = \overline{k+1,n}$, where $\bar{S}$ represents the sample mean for $S_{k+1}, \ldots, S_n$, is given by:

$$\phi(B) X_t = Z_t \ , \quad (5)$$

where $\phi(B) = 1 - \phi_1 B - \phi_{l_1} B^{l_1} - \phi_{l_2} B^{l_2} - \phi_{l_3} B^{l_3}$, $\{Z_t\} \approx \text{WN}(0,\sigma^2)$, while the coefficients ϕ_j and the variance σ^2 are calculated using the Yule–Walker equations described in [14]. From (4) and (5) we obtain the relationship:

$$\xi(B) Y_t = \phi(1)\bar{S} + Z_t \ , \quad (6)$$

where $\xi(B) Y_t = \psi(B)\phi(B) = 1 + \xi 1 B + \ldots + \xi_{k+l_3} B^{k+l_3}$.

From the following recursion relation we can determine the linear predictors $\hat{Y}_{n+h}$ for $n > k + l_3$:

$$P_n Y_{n+h} = -\sum_{j=1}^{k+l_3} \xi_j P_n Y_{n+h-j} + \phi(1)\bar{S} \quad h \ge 1 \ , \quad (7)$$

with the initial condition $P_n Y_{n+h} = Y_{n+h}$ for $h \le 0$.

C. Holt–Winters algorithm

The Holt–Winters forecasting algorithm is an exponential smoothing method which uses a set of recursions to predict the future value of series containing a trend. The main advantage of this algorithm is its simplicity, the reduced computational demand and the accuracy of the forecasts [1].

If the time series has a trend, then the forecast function is:

$$\hat{Y}_{n+h} = P_n Y_{n+h} = \hat{a}_n + \hat{b}_n h \ , \quad (8)$$

where $\hat{a}_n$ and $\hat{b}_n$ are the estimates of the level of the trend function and the slope respectively. These are calculated using the following recursive equations:

$$\begin{cases} \hat{a}_{n+1} = \alpha Y_{n+1} + (1-\alpha)(\hat{a}_n + \hat{b}_n) \\ \hat{b}_{n+1} = \beta(\hat{a}_{n+1} - \hat{a}_n) + (1-\beta)\hat{b}_n \end{cases} , \quad (9)$$

where $\hat{Y}_{n+1} = P_n Y_{n+1} = \hat{a}_n + \hat{b}_n$ represents the one-step forecast. The initial conditions are set to $\hat{a}_2 = Y_2$ and $\hat{b}_2 = Y_2 - Y_1$. The smoothing parameters α and β can be chosen either randomly (between 0 and 1), or by minimizing the sum of squared one-step errors $\sum_{i=3}^{n}(Y_i - P_{i-1}Y_i)^2$ [14].

The Holt–Winters Seasonal (HWS) algorithm extends HW to predict data which is characterized both by trend and seasonal variation with period d. The forecast function can be expressed as

$$P_n Y_{n+h} = \hat{a}_n + \hat{b}_n h + \hat{c}_{n+h} \ , \quad (10)$$

where $\hat{a}_n$, $\hat{b}_n$ and $\hat{c}_n$ are the estimates of the trend level, trend slope and seasonal component, being given by the following recursions:

$$\begin{cases} \hat{a}_{n+1} = \alpha(Y_{n+1} - \hat{c}_{n+1-d}) + (1-\alpha)(\hat{a}_n + \hat{b}_n) \\ \hat{b}_{n+1} = \beta(\hat{a}_{n+1} - \hat{a}_n) + (1-\beta)\hat{b}_n \\ \hat{c}_{n+1} = \gamma(Y_{n+1} - \hat{a}_{n+1}) + (1-\gamma)\hat{c}_{n+1-d} \end{cases} , \quad (11)$$

with the initial conditions $\hat{a}_{d+1} = Y_{d+1}$, $\hat{b}_{d+1} = (Y_{d+1} - Y_1)/d$ and $\hat{c}_i = Y_i - (Y_1 + \hat{b}_{d+1}(i-1))$ for $i = \overline{1, d+1}$. The parameters α, β and γ can take values in the range from 0 to 1 and are either chosen arbitrary or obtained after the minimization of the sum of squared one-step errors.

III. Neural Networks for Traffic Prediction

Neural Networks (NN) are widely used in the process of modeling and predicting network traffic because they can learn complex patterns through their strong self-learning and self-adaptive capabilities. NNs are able to estimate almost any function in an efficient and stable manner, when the underlying data relationships are unknown [10]. The NN model is a nonlinear, nonparametric, adaptive modeling approach which, unlike the techniques presented in Section II, relies on the observed data rather than on an analytical model [4]. The architecture and the parameters of the NN are determined solely by the dataset. NNs are characterized by nonlinear mapping and generalization ability, robustness, fault tolerance, adaptability, parallel processing ability, etc.

A neural network consists of interconnected nodes, called neurons, every connection being characterized by a weight. NN comprises several layers of neurons: *a)* an input layer, *b)* one or more hidden layers and *c)* an output layer. The most popular NN architecture is feed-forward in which the information travels through the network only in the forward direction: from the input layer towards the output layer, as illustrated in Fig. 1.

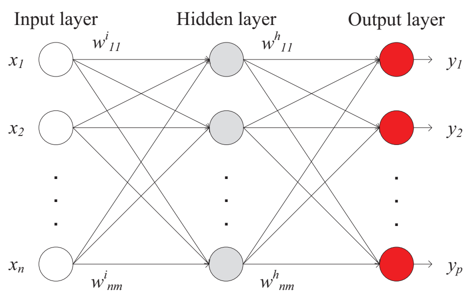


Fig. 1. Neural Network

Using a NN as a predictor involves two phases: *a)* the training phase and *b)* the prediction phase. In the training phase, the training set is presented at the input layer and the parameters of the NN are dynamically adjusted to achieve the desired output value for the input set. The most commonly used learning algorithm is the backpropagation algorithm, based on the backward propagation of the error, where the weights are changed continuously until the output error falls below a preset value. In this way, the NN can learn correlated patterns between input sets and the corresponding target values. The prediction phase represents the testing of the NN. A new input (not included in the training set) is presented to the NN and the output is calculated, thereby predicting the outcome of new input data.

The number of hidden layers and the number of nodes in each layer is usually chosen empirically. To be able to predict nonlinear values, NN must have at least one hidden layer. Too many hidden layers slow down the training process and increase the complexity of the network. In order to improve the nonlinearity of the solution, the activation functions of neurons in the hidden layer are sigmoid functions, while the output nodes have linear transfer functions.

A. Multi-task Learning

NN predictors applying the traditional single-task learning (STL) approach have only one output node and they focus on a single main task, i.e. predicting x_{t+1} based on $\{x_1, x_2, \ldots, x_t\}$. In this way, the information hidden in other tasks is neglected, such as the relationship between the historical data and x_{t+2}, although both tasks belong to the same dataset. In order to improve the generalization performance of NNs, the multi-task learning (MTL) paradigm is introduced. This means that we have a main task which is trained simultaneously with extra tasks, sharing the hidden layer of the NN, as shown in Fig. 2. By learning multiple tasks simultaneously, the NN can achieve better prediction accuracy.

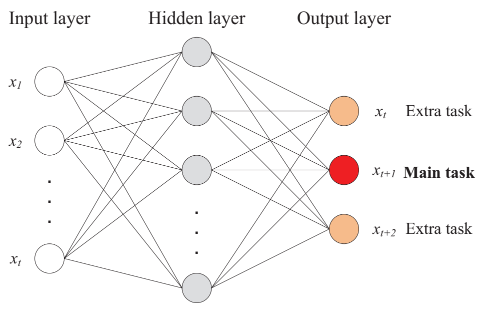


Fig. 2. NN predictor with multi-task learning

We can have one or more extra tasks, depending on the assumed complexity of the NN topology. For time series forecasting through the MTL concept, usually two extra tasks are chosen, namely the prediction of x_t and x_{t+2}, which are closely related to the main task x_{t+1}, as in [9] and [10].

B. Multiresolution Learning

In case of traditional learning, only a single representation of the training data is used in the learning process, namely the finest resolution of the original dataset. The multiresolution learning (MRL) paradigm applied to NNs exploits the correlation structures at lower resolutions of the training data. Thereby, the NN will have a better generalization capacity and the training process will be more efficient and robust, resulting in improved prediction accuracy.

Multiresolution decomposition is performed with the help of the wavelet transform. In this paper, the Haar wavelet is used. The signal s^i can be decomposed into approximation s^{i-1}, using a low-pass filter L, and detail d^{i-1}, using a high-pass filter H:

$$\begin{cases} s^{i-1} = Ls^i \\ d^{i-1} = Hs^i \end{cases} . \qquad (12)$$

The approximation s^{i-1} contains half as many samples as s^i. Reconstructing s^i means that $s^i = s^{i-1} \langle + \rangle d^{i-1} = L^* s^{i-1} + H^* d^{i-1}$, where $\langle + \rangle$ is the reconstruction operator, while L^* and H^* are low-pass and high-pass synthesis filters.

Using the decomposition algorithm described above, the original signal s^m is decomposed into a low frequency approximation s^{m-j} and high frequency details $d^{m-1}, \cdots, d^{m-j}$ at different levels, where j represents the decomposition level. For example, Fig. 3 illustrates the decomposition of the original signal s^m at approximation level 3 which can be expressed as $s^m = s^{m-3} \langle + \rangle d^{m-1} \langle + \rangle d^{m-2} \langle + \rangle d^{m-3}$.

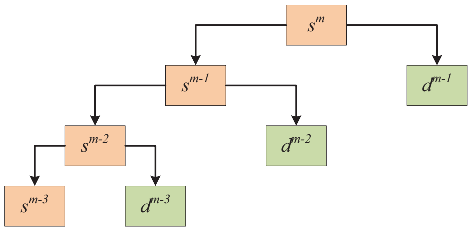


Fig. 3. Example of multiresolution decomposition

MRL means that the traditional training is decomposed in several stages, each involving a dataset of a certain resolution. The basic idea is to train the NN with the coarsest resolution s^{m-j}, followed by more finer ones and finally learning the original resolution s^m of the dataset. The first training stage starts with learning the coarsest resolution, which represents the simplest learning activity. In this stage, the NN parameters are initialized randomly. Each following stage uses the weights obtained in the previous stage and recalculates them.

Because s^i contains fewer samples than s^m, it has to be reconstructed so that both have the same length. Therefore, the training data used in each stage i will be obtained by setting the details to zero and reconstructing s^i as follows: $s^i \langle + \rangle 0^i \langle + \rangle 0^{i+1} \langle + \rangle \ldots \langle + \rangle 0^{m-1}$. Only this way can we ensure a smooth transition between the different learning stages which allows to use the same NN with the same topology in every stage.

In this paper, a decomposition level of $j = 2$ is used. The basic scheme of the corresponding NN predictor with multiresolution learning is shown in Fig. 4. The neural network can use either single-task or multi-task learning.

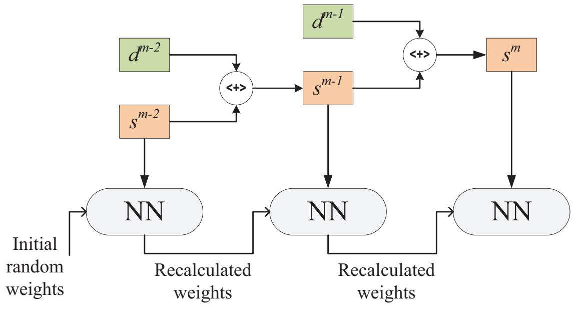


Fig. 4. NN predictor employing multiresolution learning

In the literature, neural networks with multiresolution learning are used only for predicting network traffic associated with a single variable-bit-rate (VBR) MPEG video stream, as in [11], [12], [13]. These traffic traces are characterized by a seasonal component due to the periodically sent intra-coded video frames (I-frames).

The novelty of this article is to evaluate the performance of this approach applied to aggregated Ethernet traffic traces which lack of periodical components and are known to be heavily nonlinear. In addition, we introduce a new NN training paradigm, namely the combination of the multiresolution learning with multi-task training. We intend to investigate if this combination improves the overall prediction accuracy of the NN predictor.

IV. Performance Metrics

To quantitatively assess the overall performance of the analyzed prediction methods, the following performance metrics are used to estimate the prediction accuracy:

1) MSE (Mean Square Error) is a scale dependent metric which quantifies the difference between the forecasted values and the actual values of the quantity being predicted by computing the average sum of squared errors:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \ , \tag{13}$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value and N represents the total number of predictions.

2) NMSE (Normalized Mean Square Error) can be expressed as MSE divided by the variance of the predicted time series:

$$NMSE = \frac{1}{\sigma^2} \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \ , \tag{14}$$

where σ^2 denotes the variance of the observed values during the prediction interval and is given by (15) where $\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i$ represents the mean value.

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (y_i - \bar{y})^2 \tag{15}$$

For a perfect prediction we obtain $NMSE = 0$. If $NMSE = 1$, the predictor statistically forecasts the average value of the observed data. In case of $NMSE > 1$, the performance of the prediction is worse than forecasting the mean [11].

3) MAPE (Mean Absolute Percentage Error) is a metric widely used to evaluate prediction precision. MAPE calculates the prediction error as a percentage of the observed value. Expressed in percentage terms, it presents the advantage of being easy to interpret.

$$MAPE = \frac{1}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i} \times 100\% \tag{16}$$

4) Coefficient of correlation (r) indicates the degree of association between two variables, being a measure of linear dependence. The linear correlation coefficient is sometimes referred to as the Pearson product–moment correlation coefficient (PMCC) and is defined as:

$$r = \frac{COV(Y,\hat{Y})}{\sigma_Y \sigma_{\hat{Y}}} \ , \tag{17}$$

where σ_Y and $\sigma_{\hat{Y}}$ indicate the standard deviation of the observed and the predicted values, given by (18); $COV(Y,\hat{Y})$ is the covariance between Y and $\hat{Y}$.

$$\sigma_Y = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{18}$$

The covariance is used to determine the relationship between two datasets and is obtained as follows:

$$COV(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) \ . \tag{19}$$

Values for the Pearson correlation coefficient range between -1 and 1. If $r = 1$, there is a perfect positive correlation between the actual and the predicted values, whereas $r = -1$ indicates a perfect negative correlation. If $r = 0$, we have a complete lack of correlation among the datasets.

5) Coefficient of efficiency (E):

$$E = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{20}$$

The efficiency coefficient can take values in the domain $(-\infty, 1]$. If $E = 1$, we have a perfect fit between the observed and the forecasted data. A value of $E = 0$ occurs when the prediction corresponds to estimating the mean of the actual values. An efficiency less than zero, i.e. $-\infty < E < 0$, indicates that the average of the actual values is a better predictor than the analyzed forecasting method. The closer E is to 1, the more accurate the prediction is.

V. Experimental Results

In this section the experimental results are presented and discussed. For illustration purposes, only the most interesting results will be described. The goal of the experiments is to evaluate and to compare the performance of the prediction approaches presented in Sections II and III. We intend to identify the best forecasting method for network traffic prediction, taking into account the accuracy but also the complexity of the solutions. To assess the prediction performance, the metrics described in Section IV are used.

The ARMA model and the ARAR, HW and HWS algorithms were simulated using ITSM 2000, version 7 (Student). The NN predictors were implemented and tested in Matlab using the Neural Networks Toolbox and Wavelet Toolbox.

A real-world time series of 200 consecutive traffic load measurements was used for modeling/training the predictor, and the subsequent 20 values (not included in the training set) were used for evaluating the traffic prediction performance. The quantity we are predicting is traffic load, given in bits per second [bps]. The measurements represent the average transfer rate on a 10GE (Gigabit Ethernet) link between Atlanta and Washington, measured every 10 seconds. The used trace and others are publicly available at [15]. Fig. 5 illustrates the dataset used for training and testing, along with the lower resolutions of the training set used for MRL. For testing, only the finest resolutions is needed.

The experiments were repeated for several other traffic traces, obtaining similar results. In the following, we only discuss the results obtained for this particular dataset, due to lack of space.

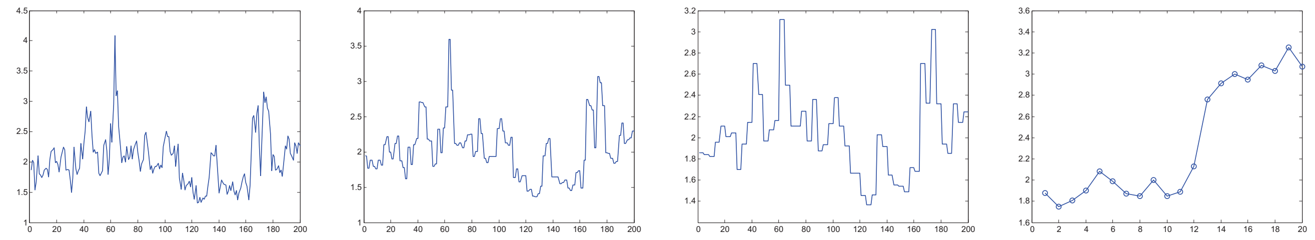
A. *Traditional Predictors*

The training data is modeled by the following ARMA$(3,5)$ process: $X(t) = 2.142X(t-1) - 2.038X(t-2) + 0.8036X(t-3) + Z(t) - 1.258Z(t-1) + 0.8077Z(t-2) + 0.3543Z(t-3) - 0.3987Z(t-4) + 0.1565Z(t-5)$, where the variance of the white noise with zero mean is $\sigma^2 = 0.050751$.

The ARAR algorithm determined the following AR(11) model: $X(t) = -0.0481X(t-1) - 0.204X(t-2) - 0.2266X(t-6) - 0.1856X(t-11)$.

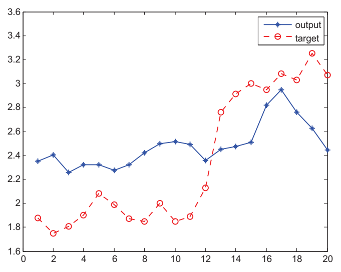
The HW algorithm predicts the testing set using (8) and (9) with the smoothing parameters: $\alpha = 1$ and $\beta = 0.04$. The HWS algorithm uses (10) and (11) for forecasting with the smoothing values: $\alpha = 0.85$, $\beta = 0$ and $\gamma = 1$.

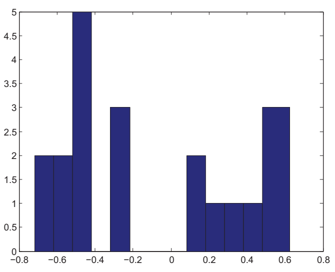
Table I compares the performance metrics of the above mentioned predictors. As can be observed from the table, the HWS algorithm presents the overall best performance among the linear predictors, having the lowest MSE and NMSE, and



(a) Original training data (b) Decomposition level 1 (c) Decomposition level 2 (d) Testing data

Fig. 5. Training and testing data [Gbps]


(a) Prediction


(b) Histogram of prediction errors

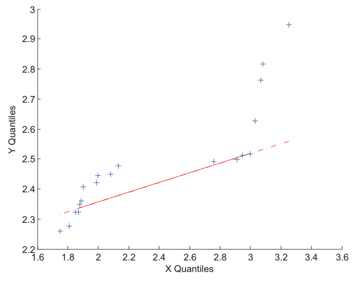

(c) Q-Q plot

Fig. 6. Prediction evaluation: Holt–Winters Seasonal algorithm

TABLE I
PERFORMANCE METRICS FOR TRADITIONAL FORECASTING MODELS

Method	MSE	NMSE	MAPE	*r*	*E*
ARMA	0.3864	1.2281	19.09%	-0.521	-0.293
ARAR	0.3068	0.975	20.08%	0.254	-0.0263
HW	0.269	0.855	20.73%	0.8935	0.2979
HWS	0.2112	0.671	19.69%	0.6933	0.3752

the highest E. Fig. 6(a) illustrates the values predicted with the HWS algorithm, compared to the observed data. Although this linear predictor has the best accuracy, the predicted data does not follow the evolution of the target values. Fig. 6(b) shows the histogram of prediction errors. The histogram lets us investigate the distribution of errors (i.e. the difference between the actual and the forecasted values): the more narrow it is, the better the prediction accuracy is. The errors are in the range $[-0.7, -0.2] \cup [0.1, 0.6]$. The Q–Q plot (Quantile–Quantile) in Fig. 6(c) compares two probability distributions as parametric curves, the parameter being the interval for the quantile. The figure displays the quantiles of the observed values (on $0x$) against the quantiles corresponding to the predicted values (on $0y$). If the samples would come from the same distribution, the plot would be linear. But this is not the case, thus we can affirm that the target values are not modeled with sufficient precision.

In order to quantify the prediction performance improvement from method a to method b in terms of MSE, the following metric is used, as in [10]:

$$\omega_{a,b} = \frac{MSE_b - MSE_a}{MSE_b} \times 100\% \, . \tag{21}$$

We denote the prediction models with the following numbers: 1. $\mapsto$ ARMA; 2. $\mapsto$ ARAR; 3. $\mapsto$ HW; 4. $\mapsto$ HWS. Table II compares the performance improvements of the traditional predictors (in terms of MSE) using (21) for each combination of pairs of predictors. The values in each line of the table can be interpreted as follows: by how much that certain predictor improved the prediction performance, compared to the predictors in the columns. Although the HWS algorithm presents a positive performance improvement, it still cannot be considered an efficient forecasting method because MAPE and NMSE are too high and the value of the efficiency coefficient E is too low.

TABLE II
PERFORMANCE IMPROVEMENTS BETWEEN TRADITIONAL PREDICTORS

	1.	2.	3.	4.
1.	0%	−25.95%	−43.64%	−82.95%
2.	2.06%	0%	−14.03%	−45.27%
3.	30.38%	12.32%	0%	−27.37%
4.	45.34%	31.16%	21.49%	0%

B. NN Predictors

Four types of NN predictors are compared: STL (Single-Task Learning), STL with MRL (Multiresolution Learning), MTL (Multi-Task Learning) and MTL with MRL. In addition, in the case of STL we compare the performance of single-step versus multi-step prediction.

In the experiments, we use NNs with a small topology in order to reduce the overall complexity of the predictor. The order of complexity for training a single epoch is $O(n_h \cdot n_o \cdot (n_i + 1))$ [4], where n_h, n_i and n_o represent the number of hidden, input and output nodes respectively. To find the appropriate NN architecture, n_i and n_h were varied between 3 and 10. We achieved the best results for a $4-5-n_o$ structured feedforward NN with backpropagation algorithm. The notation indicates a three-layer NN predictor having 4 input nodes, 5 hidden neurons and n_o output neurons. In the case of single-task learning we have $n_o = 1$, as can be seen in Fig. 7, whereas for multi-task learning we use $n_o = 3$, as presented in Fig. 8 (Δ represents a delay element).

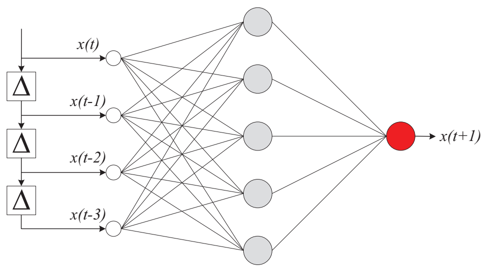


Fig. 7. Simulated NN with STL

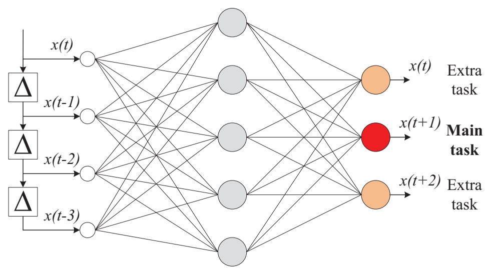


Fig. 8. Simulated NN with MTL

The learning algorithm is `trainlm` which is based on the Levenberg–Marquardt algorithm. It was chosen because it converges rapidly and offers a satisfying accuracy. The learning rate was set to 0.01. The training phase was started in each case with identical initial weights which were obtained randomly. In case of multiresolution learning, the training was conducted 100 iterations for each resolution. When no MRL was involved, the training lasted 300 epochs. We used this approach in order to ensure a comparable complexity of the different NN predictors.

In Table III we can compare the performance metrics of the NN with STL, in case of single-step and multi-step prediction.

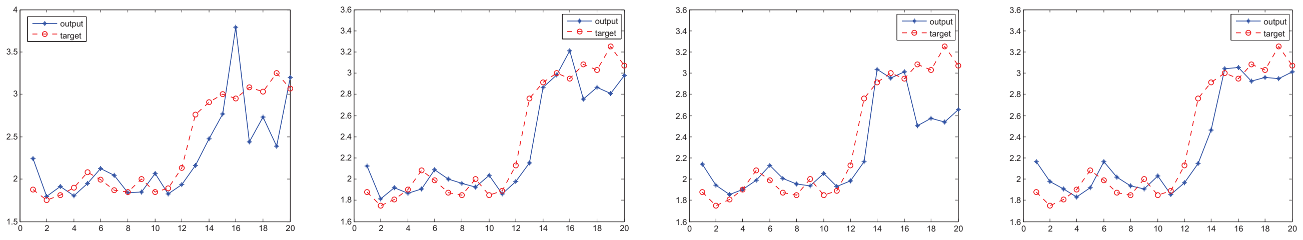
TABLE III
PERFORMANCE METRICS FOR NN WITH STL

Method	**MSE**	**NMSE**	**MAPE**	r	E
Single-step	0.14462	0.45965	11.08%	0.75583	0.51616
Multi-step	0.34069	1.0828	18.52%	0.10777	-0.13979

The multi-step prediction process presents a poor performance, having NMSE ≈ 1 and $E < 0$. This can be explained by the fact the this approach does not take into account updated input values. It uses the predicted output for a given step as an input for the next step and all other inputs are shifted back one time unit. This means that after predicting the first value, the output in Fig. 7 is connected to the $x(t)$ input. Thereby, the prediction error is propagated and the prediction accuracy constantly deteriorates. Meanwhile, single-step prediction dynamically updates the input information and the prediction accuracy is not reduced. In this case, the input node $x(t)$ can be connected to a software tool for measuring traffic load.

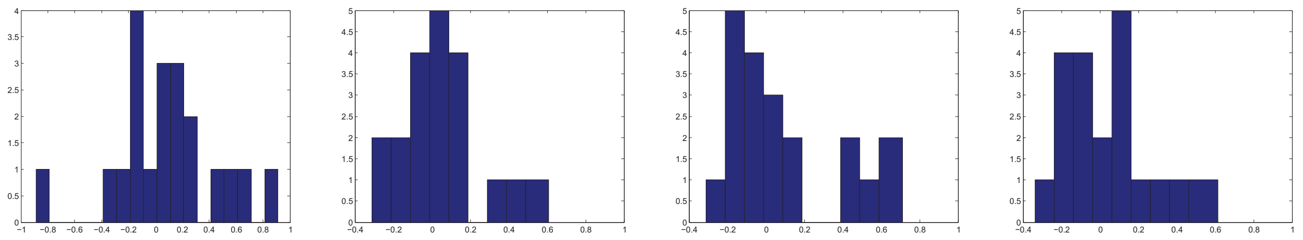
An optical comparison of the forecasted values of the network traffic predictors is possible by investigating Fig. 9. We can observe that the output values predicted with both predictors employing MRL are close to the observed values, while the predictor with STL provides the worst match between output and target values. In Fig. 10 the histograms of the NN predictors' errors are shown. The narrowest histograms are obtained for STL with MRL and MTL with MRL, the prediction error having values in the range $[-0.3, 0.6]$. As desired, we find that most error values are around 0 (between -0.2 and 0.2), as opposed to STL and MTL. Fig. 11 illustrates the different Q–Q plots. Employing MRL results in a more linear Q–Q plot, whereas the STL predictor presents a more pronounced nonlinearity.

Table IV contains the performance metrics for the four NN predictors analyzed in this paper, considering only single-step prediction. The traditional NN predictor (with STL) has the worst accuracy, but it is the simplest approach. The best results are obtained for MRL (MAPE below 7%, NMSE close to 0 and E close to 1), although the MTL with MRL has similar results, but its topology is more complex, thus it involves more calculations. Repeating the simulations several times, the results differ slightly but they are similar to this presented case.



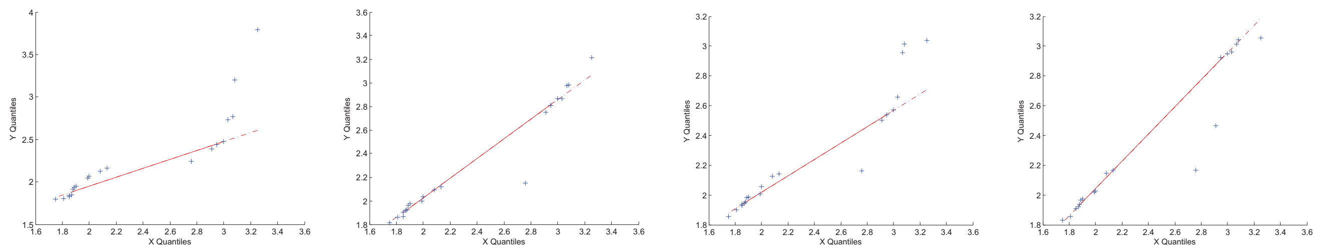
(a) STL (b) STL + MRL (c) MTL (d) MTL + MRL

Fig. 9. NN – Prediction



(a) STL (b) STL + MRL (c) MTL (d) MTL + MRL

Fig. 10. NN – Histogram of prediction errors



(a) STL (b) STL + MRL (c) MTL (d) MTL + MRL

Fig. 11. NN – Q-Q plot

TABLE IV
PERFORMANCE METRICS FOR NN PREDICTORS

Method	**MSE**	**NMSE**	**MAPE**	r	E
STL	0.14462	0.45965	11.08%	0.75583	0.51616
STL + MRL	0.049385	0.15696	6.92%	0.92009	0.83478
MTL	0.091859	0.29195	8.69%	0.86099	0.69269
MTL + MRL	0.050576	0.16074	7.53%	0.91699	0.8308

We make the following notations: 1. $\mapsto$ STL; 2. $\mapsto$ STL with MRL; 3. $\mapsto$ MTL and 4. $\mapsto$ MTL with MRL. Table V compares the performance improvement brought by different NN predictors. Positive values are obtained for STL with MRL. The additional computational complexity of MTL with MRL is not justified in terms of performance improvement. Compared to the best linear predictor, namely the HWS algorithm, the NN predictor involving STL with MRL brings a performance improvement of $\omega = 76.62\%$.

TABLE V
PERFORMANCE IMPROVEMENT BETWEEN NN PREDICTORS

	1.	2.	3.	4.
1.	0%	−192.84%	−57.44%	−185.95%
2.	65.85%	0%	46.24%	2.36%
3.	36.48%	−86.01%	0%	−81.63%
4.	65.03%	−2.41%	44.94%	0%

VI. CONCLUSIONS AND FUTURE WORK

In this paper we demonstrated that traffic load prediction is possible, with a certain accuracy. The experimental results show that nonlinear traffic prediction based on NNs outperforms linear forecasting models (e.g. ARMA, ARAR, HW) which cannot meet the accuracy requirements. If we take into account both precision and complexity, the best results are obtained by the NN predictor with multiresolution learning approach, the predicted traffic generally coinciding with the observed values. If a low computational complexity is more important, then a NN predictor with multi-task learning offers a better solutions because this approach is simpler and its performance is satisfying.

As future work we envisage to integrate the chosen predictor into a network management system and to evaluate it in real-time. Foreseeing the immediate future by employing a prediction based approach enables a proactive management.

ACKNOWLEDGMENT

This work was partly supported by the PRODOC program ”Project of Doctoral Studies Development in Advanced Technologies” (POSDRU/6/1.5/S/5 ID 7676). The authors would like to thank the Broadband Communications Research Group of the Universitat Politècnica de Catalunya for their support.

REFERENCES

[1] P. Cortez, M. Rio, M. Rocha, P. Sousa, *Internet Traffic Forecasting using Neural Networks*, International Joint Conference on Neural Networks, pp. 2635–2642. Vancouver, Canada, 2006.

[2] Z. Li, R. Wang, J. Bi, *A Multipath Routing Algorithm Based on Traffic Prediction in Wireless Mesh Networks*, Fifth International Conference on Natural Computation, Volume 6, pp. 115–119. Tianjin, China, August 2009.

[3] V. B. Dharmadhikari, J. D. Gavade, *An NN Approach for MPEG Video Traffic Prediction*, 2nd International Conference on Software Technology and Engineering, pp. V1-57–V1-61. San Juan, USA, 2010.

[4] H. Feng, Y. Shu, *Study on Network Traffic Prediction Techniques*, International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1041–1044. Wuhan, China, 2005.

[5] G. Mao, *Real-Time Network Traffic Prediction Based on a Multiscale Decomposition*, 4th International Conference on Networking, Reunion Island, France. Lecture Notes in Computer Science, Volume 3420, pp. 492–499. 2005.

[6] J. Dai, J. Li, *VBR MPEG Video Traffic Dynamic Prediction Based on the Modeling and Forecast of Time Series*, Fifth International Joint Conference on INC, IMS and IDC, pp. 1752–1757. Seoul, Korea, 2009.

[7] L. Cai, J. Wang, C. Wang, L. Han, *A Novel Forwarding Algorithm over Multipath Network*, International Conference on Computer Design and Applications, pp. V5-353–V5-357. Qinhuangdao, China, 2010.

[8] A. Abdennour, *Evaluation of neural network architectures for MPEG-4 video traffic prediction*, IEEE Transactions on Broadcasting, Volume 52, No. 2, pp. 184–192. ISSN 0018-9316, 2006.

[9] S. Sun, *Traffic Flow Forecasting Based on Multitask Ensemble Learning*, Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp. 961–964. Shanghai, China, 2009.

[10] J. Rodrigues, A. Nogueira, P. Salvador, *Improving the Traffic Prediction Capability of Neural Networks Using Sliding Window and Multi-task Learning Mechanisms*, Second International Conference on Evolving Internet, pp. 1–8. Valencia, Spain, 2010.

[11] Y. Liang, *Real-Time VBR Video Traffic Prediction for Dynamic Bandwidth Allocation*, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Volume 34, No. 1, pp. 32–47. ISSN 1094-6977, 2004.

[12] Y. Liang, X. Liang, *Improving Signal Prediction Performance of Neural Networks Through Multiresolution Learning Approach*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Volume 36, No. 2, pp. 341–352. ISSN 1083-4419, 2006.

[13] D.-C. Park, *Prediction of MPEG Traffic Data Using a Bilinear Recurrent Neural Network with Adaptive Training*, International Conference on Computer Engineering and Technology, pp. 53–57. Singapore, 2009.

[14] P. J. Brockwell, R. A. Davis, *Introduction to Time Series and Forecasting*, Second Edition. Springer-Verlag,ISBN 0-387-95351-5, 2002.

[15] Traffic measurements — http://dc-snmp.wcc.grnoc.iu.edu/i2net/#