Using Mathematics for Data Traffic Modeling within an e-Learning Platform

Marian Cristian Mihăescu
University of Craiova, Faculty of Automatics, Computers and Electronics
Software Engineering Department
Bvd. Decebal, Nr. 107, Craiova, Dolj, Romania

Abstract-e-Learning data traffic characterization and modeling may bring important knowledge about characteristics of traffic. It is considered that without measurement it is impossible to build realistic traffic models. We propose an analysis architecture employed for characterization and modeling using data mining techniques and mathematical models. The main problem is that usually real data traffic has to be measured in real time, saved and later analyzed. The proposed architecture uses data from application level. In this way the data logging process becomes a much easier task with practically almost the same outcomes.

I. INTRODUCTION

Tesys e-Learning platform was developed and deployed. It has a built in mechanism for monitoring actions performed by users and data traffic transferred during usage. In this paper we study the possibility of modeling data traffic using data mining techniques and mathematics based on performed actions. This would have great benefits regarding the overhead within the platform. Introduction presents in short Tesys e-Learning platform. In second section there are presented the employed methods: actions and data monitoring, the process of clustering users. The clustering process groups similar users based on a specific similarity function. At cluster level, self-similarity of data traffic is then examined. This is accomplished by estimating Hurst parameter.

In third section of the paper there is presented the proposed architecture and the analysis process. In short, the architecture and employed process will try to estimate the data traffic self-similarity within a cluster of users. In fourth section there are presented obtained results. Finally, the conclusions and future works are presented.

Tesys is an e-Learning platform that has been designed and implemented and was deployed in a continuous learning program. Four types of users may use the platform: sysadmin, secretary, professor and student. Secretaries and professors work together to manage the infrastructure the student will use. Secretaries manage the professors, disciplines and students. Professors take care of the assigned disciplines in terms of course materials, test and exam questions. An e-learning template mechanism creates course materials in a very attractive and interactive fashion. Professors and students communicate with each other through a dedicated module of the platform. The students are the main beneficiaries of the platform. They can download course materials take tests at each chapter of the disciplines and take the final examination. They can also communicate peer-to-peer with professors and secretaries. The sysadmin has more a supervising role since besides managing secretaries he may obtain information regarding the activity done on the platform.

The platform has built in capability of monitoring and recording user's activity at application level. The activity represents valuable data since it is the raw data for our machine learning and modeling process. User's sequence of sessions makes up his activity. A session starts when the student logs in and finishes when the student logs out. Under these circumstances, a sequence of actions makes up a session.

II. METHODS AND MATERIALS

From the design phase of the platform, there were adopted two methodologies for monitoring actions. Since the business logic of the platform is Java based, log4j utility package was employed as a logging facility and is called whenever needed within the logic of the application. The utility package is easy to use; log4j.properties properties file manages the logging process. The setup process states the logs are saved in idd.log file. The main drawback of this technique is that the data from the file is in a semi-structured form. This makes the information retrieval to be not so easy task to accomplish. On the advantages, logging activity may be very helpful in auditing the platform or even finding security breaches. This logging facility is also very helpful when debugging during development or when analyzing peculiar behavior during deployment.

To overcome the semi-structured shape of logged activity a structured way of gathering activity information was enforced. The activity table was added in the database and all actions were recorded in the manner of one record per action. The fields of activity table are: id (primary key), user id (identifies the user who performed the action), date (stores the date when the action was performed), action (stores a tag that identifies the action), details (stores details about performed action).

The quantity of data traffic transferred by users was also measured and logged. Whenever a user made a request the size in bytes of the response was measured and saved. Still, the recorded quantities of transferred data were recorded at action level. For example, if a user performs a test composed of 20 questions this is recorded as a single data transfer. The sizes of questions are cumulated and the result represents the data transferred at the time when the test started.

There are many different ways for representing patterns that can be discovered by machine learning. From all of them we choose clustering, which is the process of grouping a set of physical or abstract objects into classes of similar objects [1]. Basically, for our platform we create clusters of users based on their activity.

As a product of clustering process, associations between different actions on the platform can easily be inferred from the logged data. In general, the activities that are present in the same profile tend to be found together in the same session. The actions making up a profile tend to co-occur to form a large item set [8].

There are many clustering methods in the literature: partitioning methods such as [9], hierarchical methods, density-based methods such as [10], grid-based methods or model-based methods. Hierarchical clustering algorithms like the Single-Link method [4] or OPTICS [7] compute a representation of the possible hierarchical clustering structure of the database in the form of a dendrogram or a reachability plot from which clusters at various resolutions can be extracted, as has been shown in [11]. From all of these we chose to have a closer look on partitioning methods.

From a different perspective for a cluster there may be computed the following parameters:

$\mu = \frac{x_1 + x_2 + \ldots + x_n}{n}$, the means

$\sigma = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \ldots + (x_n - \mu)^2}{n-1}$,the standard deviation

p, the probability

The sum of all probabilities for all clusters is 1. If we know which of the distributions each instance came from, finding the parameters is easy. On the other hand, if the parameters are known finding the probabilities that a given instance comes from each distribution is easy. Given an instance x, the probability that it belongs to cluster A is:

$$\Pr[A \mid x] = \frac{\Pr[x \mid A] - \Pr[A]}{\Pr[x]} = \frac{f(x;\mu_A,\sigma_A)p_A}{\Pr[x]}$$

where $f(x;\mu_A,\sigma_A)$ is the normal distribution function for cluster A, that is:

$$f(x;\mu_A,\sigma_A) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The EM algorithm [2] takes into consideration that we know neither of these things: not the distribution that each training instance came from, nor the parameters μ, σ or the probability. So, we adopt the procedure used for the k-means clustering algorithm and iterate. Start with initial guess for the five parameters, use them to calculate the cluster probabilities for each instance, use these probabilities to reestimate the parameters, and repeat. This is called the EM algorithm for "expectation-maximization". The first step, the calculation of cluster probabilities (which are the "expected" class values) is "expectation"; the second, calculation of the distribution parameters is "maximization" of the likelihood of the distributions given the data [8].

A slight adjustment must be made to the parameter estimation equations to account for the fact that it is only cluster probabilities, not the clusters themselves that are known for each instance. These probabilities just act like weights. If w_i is the probability that instance i belongs to cluster A, the mean and standard deviation are:

$$\mu_A = \frac{w_1 x_1 + w_2 x_2 + \ldots + w_n x_n}{w_1 + w_2 + \ldots + w_n}$$

$$\sigma_A^2 = \frac{w_1(x_1 - \mu)^2 + w_2(x_2 - \mu)^2 + \ldots + w_n(x_n - \mu)^2}{w_1 + w_2 + \ldots + w_n}$$

where x_i are all the instances, not just those belonging to cluster A. Technically speaking, this is a "maximum likelihood" estimator of the variance.

Now we shall discuss about how iterating process is terminated. The k-means algorithm stops when the classes of instances don't change from one iteration to the next – a "fixed point" has been reached. In the EM algorithm things are not quite so easy: the algorithm converges toward a fixed point but never actually gets there. But we can see how close it is by calculating the overall likelihood that the data came from this dataset, given the values for the parameters (mean, standard deviation and probability). This overall likelihood is obtained by multiplying the probabilities of the individual instances i:

$$\prod_i \left(p_A \Pr[x_i \mid A] + p_B \Pr[x_i \mid B]\right)$$

where the probabilities given the clusters A and B are determined from the normal distribution function $f(x;\mu,\sigma)$.

This overall likelihood is a measure of the "goodness" of clustering and increases at each iteration of the EM algorithm. Again, there is a technical difficulty with equating the probability of a particular value of x with $f(x;\mu,\sigma)$, and in this case the effect does not disappear because no probability normalization operation is applied. The upshot is that the likelihood expression above is not a probability and does not necessarily lie between zero and one: nevertheless, its magnitude still reflects the quality of the clustering. In practical implementations its logarithm is calculated instead: this is done by summing the logs of individual components, avoiding all the multiplications. But the overall conclusion still holds: iterate until the increase in log-likelihood becomes negligible. For example, a practical implementation might iterate until the difference between successive values of log-likelihood is less than 10^{-10} for ten successive iterations. Typically, the log-likelihood will increase very sharply over the first few iterations and then converge rather quickly to a point that is virtually stationary.

Although the EM algorithm is guaranteed to converge to a maximum, this is a local maximum and may not necessarily be the same as the global maximum. For a better chance of obtaining the global maximum, the whole procedure should be repeated several times, with different initial guess for the parameter values. The overall log-likelihood figure can be used to compare the different final configuration obtained: just choose the largest of the local maxima [8].

The EM algorithm is implemented in Weka package[12] and needs the input data to be in a custom format called *arff.*

Self-similarity and long-range dependence of data traffic are discussed in detail in [13, 14 and 15]. A process is considered to be self-similar if Hurst parameter satisfies the condition:

$$Y(t) = a^{-H} Y(at) \quad t > 0, a > 0, 0 < H < 1$$

where the equality is in the sense of finite-dimensional distributions. A second definition of self-similarity that is more appropriate in the context of standard time series, involves a stationary sequence $X = \{X(i), i > 1\}$. Let

$$X^{(m)}(k) = (1/m) \sum_{i=(k-1)m+1}^{km} X(i), \quad k = 1,2,\ldots.$$

It is not possible to use the definition to check whether a finite traffic trace is self-similar or not. Instead different features of self-similarity such as slowly decaying variances are investigated in order to estimate the Hurst parameter H.

Parameter H can take any value between 1/2 and 1 and the higher the value the higher the degree of self-similarity. For smooth Poisson traffic the value is H=0.5. There are four methods are used to test for self-similarity. These four methods are all heuristic graphical methods, they provide no confidence intervals and they may be biased for some values of H. The rescaled adjusted range plot (R/S plot), the Variance-Time plot and the Periodogram plot, and also the theory behind these methods, are described in detail by Beran [13] and Taqqu et al. [15]. Molnar et al. [16] describes the index of dispersion for counts method and also discuss how the estimation of the Hurst parameter can depend on estimation technique, sample size, time scale and other factors.

III. Proposed Analysis Process

The analysis process starts from logged data about actions and data traffic and comes up with an estimation of Hurst parameter. This estimation of self-similarity represents important knowledge in characterizing and modeling data traffic. In Figure 1 it is presented the employed analysis process.

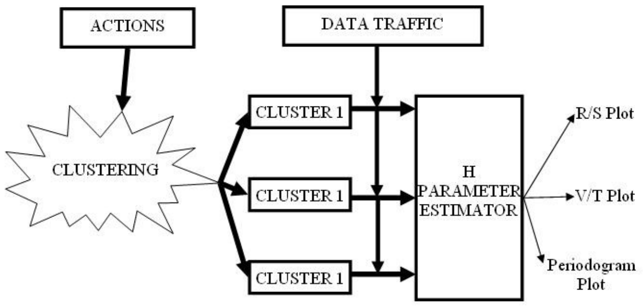

Figure 1. Analysis process.

The analysis process starts by creating clusters of users based on their activity. For this there is used only the data regarding performed actions. Once the clusters of users are obtained, the data traffic transferred within each cluster is taken into consideration by H Parameter Estimator module. This module will produce three plots: R/S plot, Variance-Time plot and the Periodogram plot.

The R/S method is one of the oldest and most well known methods for estimating H. Let *Xt* denote the number of packets that arrive at time *t*, i.e the number of packets in bin *t*.

$$Y_j = \sum_{i=1}^{I} X_i$$

represents the cumulative inflow up to time j. The R/S-statistic or rescaled adjusted range is defined by the ratio:

$$R/S = \frac{R(t,k)}{S(t,k)} \text{ where}$$

$$R(t,k) = \max_{0\le i\le k}\left[Y_{t+i} - Y_t - \frac{i}{k}(Y_{t+k} - Y_t)\right] - \min_{0\le i\le k}\left[Y_{t+i} - Y_t - \frac{i}{k}(Y_{t+k} - Y_t)\right]$$

is called the adjusted range and

$$S(t,k) = \sqrt{k^{-1} \sum_{i=t+1}^{t+k} \left(X_i - \overline{X}_{t,k}\right)^2} \text{ where}$$

$\overline{X}^{t,k} = k^{-1} \sum_{i=t+1}^{t+k} X_i$ makes it possible to study properties that are independent of scale.

To determine the Hurst parameter H the ratio R/S is calculated for every possible, or a sufficient number of, values of t and k and log R/S is plotted against log k. The slope of a straight line fitted to the points in the plot, for instance by the least square method, is an estimation of the parameter H.

For variance-time plot, firstly the mean of each pair of consecutive, non-overlapping bins are calculated and then the variance of these means is calculated. The 2-logarithm of the variance is plotted against the logarithm of the block size i.e 1. Then the same thing is done for blocks of size 4,8,16,..,length(X)/2 bins. The parameter H can be estimated by fitting a simple least squares line through the resulting points and using the relation slope = 2H - 2.

The periodogram is defined as:

$$I(\gamma) = \frac{1}{2\pi N} \left| \sum_{j=1}^{N} X(j) e^{ij\gamma} \right|^2$$

where γ is a frequency, N is the length of the series, and X is the time series. I(γ) is an estimator of the spectral density. A series with long-range dependence should have a periodogram which is proportional to $|\gamma|^{1-2H}$ close to the origin. An estimation of H is given by fitting a straight line to a log-log plot of the periodogram against the frequency.

The slope of the line is approximately 1-2H. In practice only the lowest 10% of the roughly N/2 frequencies is used when estimating H [15].

IV. Results

The EM algorithm is implemented in Weka package[18] and needs the input data to be in a custom format called *arff*. Under these circumstances we have developed an offline Java application that queries the platform's database and crates the input data file called *activity.arff*. This process is automated and is driven by a properties file in which there is specified what data will lay in activity.arff file.

The most important step in this procedure is the attribute selection and the granularity of their nominal values. The number of attributes and their meaning has a crucial importance for the whole process since irrelevant attributes may degrade classification performance in sense of relevance. On the other hand, the more attributes we have the more time the algorithm will take to produce a result. Domain knowledge and of course common sense are crucial assets for obtaining relevant results.

For a student in our platform we may have a very large number of attributes. Still, in our procedure we used only three: the number of loggings, the number of taken tests and the number of sent messages. Here is how the *arff* file looks like:

@relation activity

@attribute noOfLogins
{<10,<50,<70,<100,>100}
@attribute noOfTests
{<10,<20,<30,<50,>50}
@attribute noOfSentMessages
{<10,<20,<30,<50,>50}

@data
<50,<10,<10,
>100,<20,<20,

As it can be seen from the definition of the attributes each of them has a set of five nominal values from which only one may be assigned. The values of the attributes are computed for each of the 375 students and are set in the *@data* section of the file. For example, the first line says that the student logged in less than fifty times, took less than ten tests and sent less than ten messages to professors.
The granularity for the nominal values of the attributes can be also increased. In our study we considered only five possible values but we can consider testing the algorithm with more possible values. This should have great impact on the number of obtained clusters. The time taken by the algorithm to produce results should also increase.
In order to obtain relevant results we pruned noisy data. We considered that students for which the number of loggings, the number of taken tests or the number of sent messages is zero are not interesting for our study and degrade performance and that is why all such records were deleted. After this step there remained only 268 instances.

Running the EM algorithm created three clusters. The procedure clustered 91 instances (34%) in cluster 0, 42 instances (16%) in cluster 1 and 135 instances (50%) in cluster 3. The final step is to check how well the model fits the data by computing the likelihood of a set of test data given the model. Weka measures goodness-of-fit by the logarithm of the likelihood, or log-likelihood: and the larger this quantity, the better the model fits the data. Instead of using a single test set, it is also possible to compute a cross validation estimate of the log-likelihood. For our instances the value of the log-likelihood is -2.61092 which represent a promising result in the sense that instances (in our case students) may be classified in three disjoint clusters based on their activity.

The clustering process produced the following results:

TABLE I
DISTRIBUTION OF USERS IN CLUSTERS

Cluster	No. of users
0	91 (34%)
1	42 (16%)
2	135 (50%)

For obtained clusters a study of self similarity of traffic was performed. More precisely, self-similarity was studied for cluster 0 formed of 91 students (34%).

In 6 month of functioning on the platform there were executed over 10,000 actions of different types: course downloads, messaging, self tests, exams. For computations a packet was considered to have 1,000 bytes.

For estimation of Hurst parameter there was chosen a 3 hours interval, between 18:00 and 21:00 which is considered to be a heavy traffic period. This may be observed from the general traffic statistics presented in Figure 2.

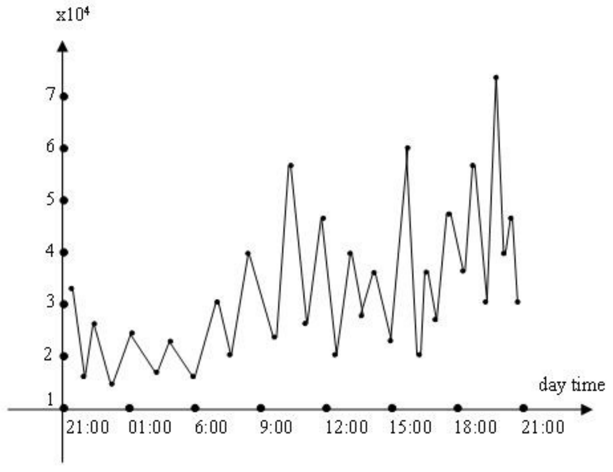

Figure 2. General data traffic on e-Learning platform

The interval from 18:00 to 21:00 was chosen for close analysis. The R/S plot estimated H parameter to a value of 0.89. The time-variance plot showed a slope of -0.320 which means a value of H of 1+slope/2=0.84. The IDC (Index of Dispersion for Counts) shows an H parameter of 0.88. In Periodogram plot there may be observed a value of H = 0.85. These methods do not obtain exactly the same values but values are over 0.5 which is a good indication of traffic's self-similarity.

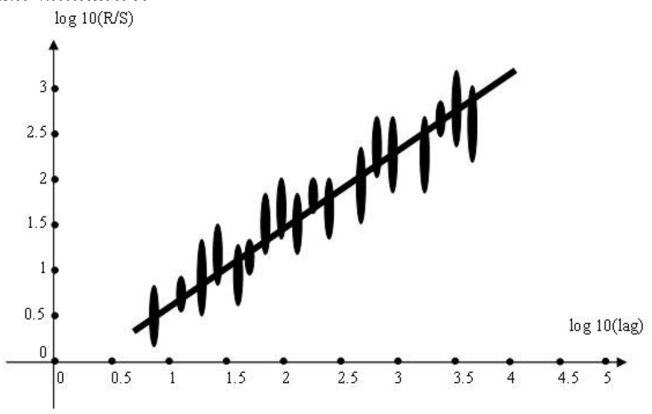

Figure 3. Estimation of Hurst parameter – R/S plot

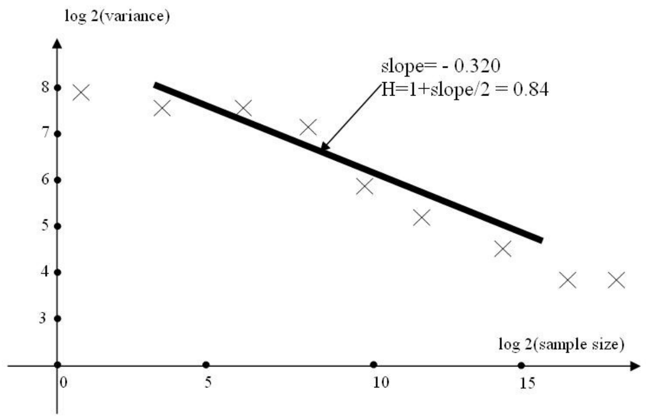

Figure 4. Estimation of Hurst parameter – Variance-Time plot

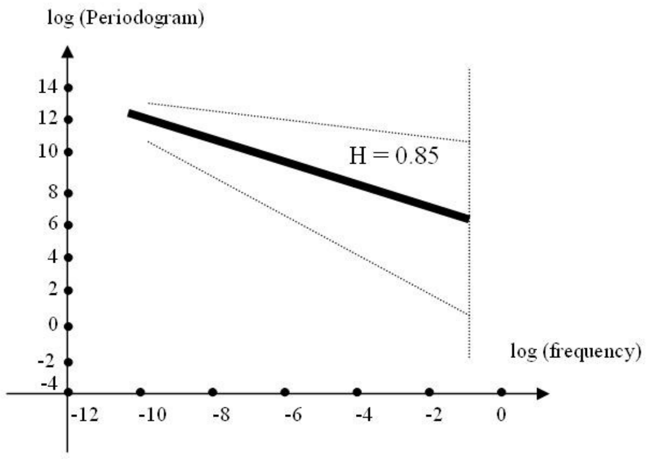

Figure 5. Estimation of Hurst parameter – Periodogram plot

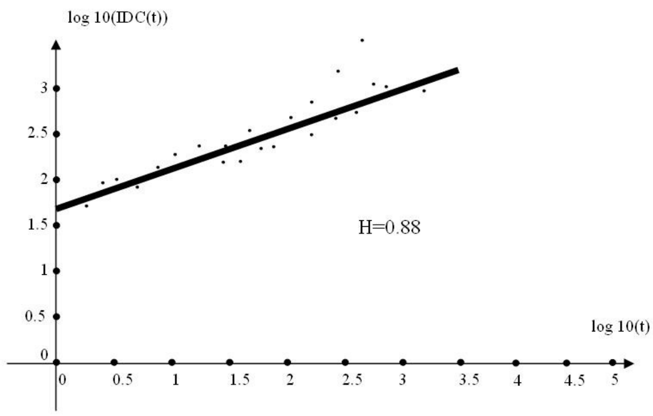

Figure 6. Estimation of Hurst parameter – IDC plot

The self-similarity of byte traffic presents similar values for H parameter. The number of bytes transferred in each bin were computed and results presented in Table II. In this table there are presented estimations of H parameter for different dimensions of time bin.

TABLE II
HURST PARAMETER ESTIMATES FOR 18:00-21:00 TIME INTERVAL

Bin size	**Packets**			**Bytes**		
	$H_{R/S}$	H_{VT}	H_P	$H_{R/S}$	H_{VT}	H_P
2h	0.85	0.84	0.87	0.83	0.86	0.88
4h	0.82	0.78	0.85	0.81	0.86	0.88
6h	0.84	0.83	0.89	0.77	0.79	0.85

Having in mind that non-stationary traffic may be easily taken as self-similar stationary traffic there were also examined smaller intervals of time bins. H parameter was estimated for each of the 6 intervals of 30 minutes between 18:00 and 21:00. The results are presented in Figure 7.

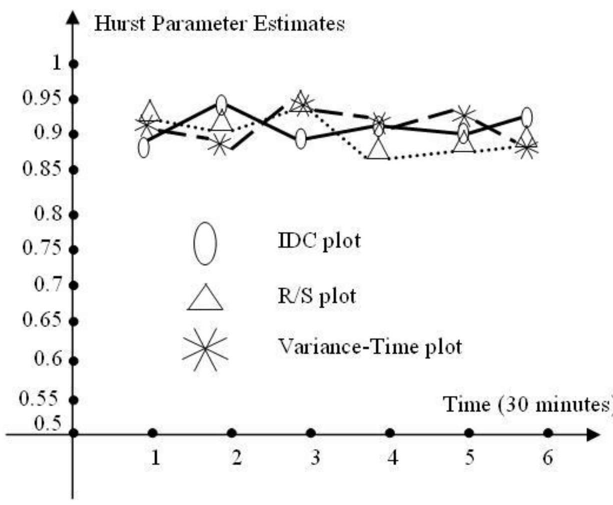

Figure 7. Estimation of Hurst parameter for intervals of 30 minutes from 18:00 to 21:00

In this way, there was estimated H parameter for three hours from a complete interval of 24 hours. Estimation of H parameter for other intervals is presented in Figure 8.

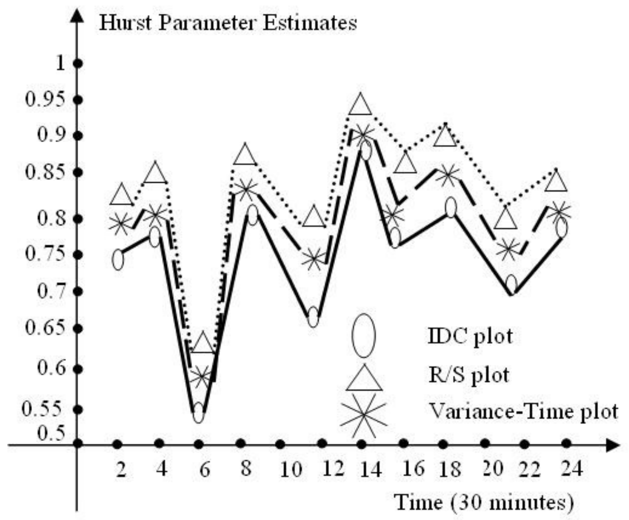

Figure 8. Estimation of Hurst parameter for each hour of traffic monitoring

Estimations were accomplished for packet data traffic and a time interval of 15 minutes. All three methods show high values between 15:00 and 20:00. Because this time interval corresponds to moments when the platform was intensely used confirms the researches of Leland et. al. [17] that expressed the idea that when network load is high than the degree of self-similarity is increased.

The fact that traffic is found to be self-similar does not change its behavior but it changes the knowledge about real traffic and also the way in which traffic is modeled. It has lead many [19] to abandon the Poisson-based modeling of network traffic for all but user session arrivals. Real traffic, well described as self-similar, has a "burst within burst" structure that cannot be described with the traditional Poisson-based traffic modeling.

V. CONCLUSIONS

Tesys e-Learning platform was developed and deployed. The platform has built in capability of logging actions performed by users and data traffic quantities transferred among users. After six month of usage, a large quantity of data was available to analyze.

Data analysis is done using EM clustering algorithm implemented by Weka system and mathematical traffic modeling.

The user clustering process produced three clusters of users. The mathematical traffic modeling was performed on data obtained for users that belong to a certain cluster.

Mathematical modeling estimates the self-similarity of data traffic. This is accomplished by heuristic graphical methods: R/S plot, variance-time plot, IDC plot, periodogram plot. The analysis is performed rigorously for a three hours interval, from 18:00 to 21:00 but also for the whole day.

All the analysis follows a proposed analysis process that has as input data regarding executed actions and transferred bytes within the platform and has as output estimates of the Hurst parameter.

Values found for Hurst parameter are very promising. All calculations showed values above 0.7 and many times above 0.8 which indicate a good level of self-similarity.

The differences regarding Hurst parameter are due to estimation method, bin size and point of time.

As future works it would be interesting to analyze if there is a correlation between the user behavior and the amount of data transferred. Since employed clustering process partitioned users according with their behavior this paper is a step in showing that self-similarity in network traffic can be explained in terms of user behavior.

REFERENCES

[1] Jiawei Han, Micheline Kamber "Data Mining – Concepts and Techniques" Morgan Kaufmann Publishers, 2001.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," Proc. of the 20th VLDB Conference, pp. 487-499, Santiago, Chile, 1994.

[3] MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. In 5th Berkeley Symp. Math. Statist. Prob., 281-297, 1967.

[4] Ester M., Kriegel H.-P., Sander J., Xu X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proc. KDD'96, Portland, OR, pp.226-231,1996.

[5] Sibson, R. SLINK: An Optimally Efficient Algorithm for the Single-link Cluster Method. The Computer Journal, 16(1): 30-34, 1973.

[6] Ankerst, M., Breuing, M., Kriegel, H-P., Sander, J. OPTICS: Ordering Points to Identify the Clustering Structure. In SIGMOD'99, 49-60, 1999.

[7] Sander, J., Qin, X., Lu, Z., Niu, N, Kovarsky, A. Automated Extraction of Clusters from Hierarchical Clustering Representations. PAKDD'03.

[8] Ian H. Witten, Eibe Frank "Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations" Morgan Kaufmann Publishers, 2000.

[9] Nasraoui O., Joshi A., and Krishnapuram R., "Relational Clustering Based on a New Robust Estimator with Application to Web Mining," Proc. Intl. Conf. North American Fuzzy Info. Proc. Society (NAFIPS 99), New York, June 1999.

[10] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," Proc. of the 20th VLDB Conference, pp. 487-499, Santiago, Chile, 1994.

[11] B. Mobasher, N. Jain, E-H. Han, and J. Srivastava "Web mining: Pattern discovery from World Wide Web transactions," Technical Report 96-050, University of Minnesota, Sep, 1996

[12] Holmes, G., Donkin, A., and Witten, I.H., Weka: a machine learning workbench. Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia, pp. 357- 361., 1994.

[13] J. Beran. Statistics for Long-Memory Processes. Chapman & Hall, New York, 1994.

[13] W. Willinger, M. S. Taqqu, R. Sherman, and D. Wilson, "Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level," in Proceedings of SIGCOMM '95, pp. 100-113.

[14] W. Willinger, V. Paxson and M.S. Taqqu, "Self-similarity and heavy tails: structural modeling of network traffic," in A practical guide to heavy tails: statistical techniques and applications, R. Adler, R. Feldman and M.S. Taqqu, Eds. Birkhauser, Boston, 1998.

[15] M. S. Taqqu and V. Teverovsky, "On estimating the intensity of long-range dependence in finite and infinite variance time series," in A practical guide to heavy tails: statistical techniques and applications, R. Adler, R. Feldman and M.S. Taqqu, Eds. Birkhauser, Boston, 1998.

[16] S. Molnar, A. Vidacs and A. Nilsson, "Bottlenecks on the Way Towards Fractal Characterization of Network Traffic: Estimation and Interpretation of the Hurst Parameter". International Conference on the Performance and Management of Communication Network, Tsukuba, Japan, 17-21 November 1997.

[17] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson, "On the self-similar nature of Ethernet traffic (Extended version)," IEEE/ACM Transactions on Networking, vol. 2, no. 1, pp. 1-15, 1994.

[18] Holmes, G., Donkin, A., and Witten, I.H., "Weka: a machine learning workbench", Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia, pp. 357- 361, 1994.

[19] V. Paxson and S.Floyd, "Wide-area traffic: The failure of poisson modeling," in IEEE/ACM Transactions on Networking, vol. 3, no. 3, pp. 226-244, 1995.