A Comparative Study of Path Performance Metrics Predictors

Juan Pablo Nariño-Mendoza[(1)], Benoit Donnet[(2)], Pierre Dupont[(1)]
Université catholique de Louvain
Computing Science and Engineering Department
[(1)] Machine Learning Group – [(2)] IP Network Lab
Place Sainte-Barbe 2, B1348 Louvain-la-Neuve – Belgium *

ABSTRACT

Using quality-of-service (QoS) metrics for Internet traffic is expected to improve greatly the performance of many network enabled applications, such as Voice-over-IP(VoIP) and video conferencing. However, it is not possible to constantly measure path performance metrics (PPMs) such as delay and throughput without interfering with the network.

In this work, we focus on PPMs measurement scalability by considering machine learning techniques to estimate predictive models from past PPMs observations. Using real data collected from PlanetLab, we provide a comparison between three different predictors: AR(MA) models, Kalman filters and support vector machines (SVMs). Some predictors use delay and throughput jointly to take advantage of the possible relationship between PPMs, while other predictors consider PPMs individually. Our current results illustrate that the best performing model is an individual SVM specific to each time series. Overall, delay can be predicted with very good accuracy while accurate forecasting of throughput remains an open problem.

1. INTRODUCTION

Since the early 90's, the Internet continues to grow quickly and this evolution is a matter of concern that can lead to new methodological opportunities. Our evaluation of the situation is based on two elements: the *quality of service demand* and the *multiconnectivity*.

The Internet behavior was originally quite simple. The content (or the service) was located at a single place and each client willing to access it used a unique path. Most applications used at that time were not quality-of-service (QoS) consuming. Nowadays, the situation is strongly different. Indeed, the content and services are now replicated among a set of servers and, sometimes, directly among clients. The perfect example of such a situation is peer-to-peer (P2P). In addition, current and future applications, such as telephony and video-over-IP (VoIP), content servers, or online games, require more QoS. Various techniques have been proposed in order to add QoS to the network [1, 2]. Unfortunately, while these techniques might work at very small scales, each attempt to include them in the global Internet (and, thus, make them scalable) has failed so far.

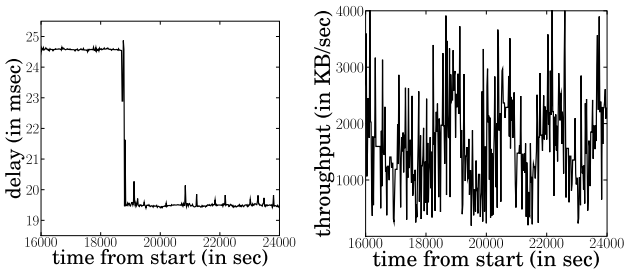
In parallel to this QoS demand, we are seeing an increasing fragmentation of the Internet, composed of roughly 20,000 corporate networks and different operators. An increasing number of corporate networks and operators are connected simultaneously to several Internet service providers (i.e., *multihoming*) or share several connections to a given provider (i.e., *multiconnectivity*) in order to benefit from redundancy in case of network failures and from a better QoS [3, 4, 5].

As a result, an application has currently several paths at its disposal for reaching a given content. Allowing any application to select the best possible path is a crucial question that follows the current discussions on the future Internet among working groups in the European Commission [6] or in the United States [7, 8]. A potential approach for path selection would be to allow each end-host to continuously measure the network (candidate metrics could be the delay, the throughput, the loss rate, the jitter, or the number of hops) and to select a path based on the end-host requirements and measurement results.

However, it is not reasonable to develop an approach that would require each end-host to perform its own set of measurements each time an application needs to select a path. The traffic generated by those measurements could be huge and can potentially be redundant, leading to scalability issues. In addition, one can encounter a security risk. Indeed, the measurement traffic might be quickly seen by a third party as a kind of intrusion. A way to fix those issues would be to aggregate measurements from well known vantage points, allowing those vantage points to collaborate in order to exchange information, and sample measurements [9]. The cost associated to such a solution is a precision loss. In the current state of the art, we are able to sample, for instance, the throughput between two isolated points of the network (see, for instance, [10]). However, applying sampling techniques to the scale of the Internet is a totally open and very complex question due to the size of the network. A better solution would be to provide a service

*This research work is partially funded by the European Commission through the ECODE project (INFSO-ICT-223936) under the European Seventh Framework Programme (FP7). Mr. Donnet's work is supported by the FNRS (Fonds National de la Recherche Scientifique, rue d'Egmont 5 – 1000 Bruxelles, Belgium.).

Advanced Learning for Networking 2009 Seattle, WA



(a) delay (b) throughput

Figure 1: Details of the times series for PPMs measured between Cambridge and Köln

for application that centralizes measurements and, based on those measurements and an application requirements, perform the path selection. However, such a solution still needs to carefully measure the network.

The measurement scalability is exactly the problem we tackle in this paper. Indeed, we investigate how machine learning techniques might be used for inferring *Path Performance Metrics* (PPMs). In particular, we are interested in predicting delay or throughput. Using a large dataset we collected on PlanetLab, we discuss several time series techniques for predicting PPMs. We are also interested in using the possible statistical dependence between throughput and delay to improve the prediction of each of them. This is a step towards the *joint prediction*, i.e., from a joint time series of throughput and delay, predict those quantities together. Machine learning approaches to PPMs prediction has already been studied in the networking research community (see for instance [11, 12, 13]) but these works rely on limited datasets based either on RTT measurements or on a controlled environment.

The remainder of this paper is organized as follows: Sec. 2 explains how machine learning techniques can be used for inferring Internet path performance; Sec. 3 discusses how we collect and process data for our experiments; Sec. 4 provides a comparison between different Internet path performance metric predictors; finally, Sec. 5 concludes this paper by summarizing its main contributions and discussing future directions.

2. LEARNING PATH PERFORMANCE METRICS

Predicting future path performance metrics from past measurements can be stated, in machine learning terms, as a *time series* regression problem. The *training* dataset D consists of measurements from the previous time step $t-1$ back to some index $t-k$ in the past:

$$D = \{y_{t-1}, y_{t-2}, \ldots, y_{t-k}\} \quad (1)$$

A model of this time series aims at predicting the next metric value y_t and, possibly, additional values in the future $y_{t+1}, y_{t+2}, \ldots$ This problem is a time series prediction problem.

It is necessary to state the statistical properties of the time series to be predicted. The most important feature of PPMs time series is its non-stationarity and non-linearity [14]. This imposes conditions over the possible models that can be used. However, as can be seen from Fig. 1 in the case of delay (Fig. 1(a)), non-stationarity is mostly due to the presence of jump discontinuities (called *interventions* in time series literature). Once the interventions are left out, the behavior, although not stationary in a strict sense, presents little variation in most cases, specially in measurements taken from well connected vantage points.

Throughput time series present a bigger challenge. Throughput, as it can be seen from Fig. 1(b), is quite non-linear and nonstationary. Throughput also presents wide variations and little stability within a given range. Since the delay and throughput measurements were taken jointly, we explore some preliminary ideas to make use of both PPMs to improve the prediction of each of them.

In this work, we compare the predictive accuracy of some linear versus non-linear time series models. The models we compared are *Auto-regressive Moving Average Models* (ARMA) [15], *Kalman Filters* [16], and *Support Vector Machines* [17] for time series.

An ARMA model uses samples from the past (i.e., dataset D) in order to construct a model for forecasting. A general $ARMA(p,q)$ model is described by the following equation:

$$y_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} + \phi_1 \epsilon_{t-1} + \cdots + \phi_1 \epsilon_{t-q} + e_t \quad (2)$$

where $\epsilon_i \sim N(0, \sigma^2)$, $e_t \sim N(0, \sigma^2)$, p indicates the order of the auto-regression, and q the order of the moving average. Preliminary experiments have shown that the moving average components do not offer improved results over simple AR models. Hence, $q = 0$ is assumed for the rest of this paper and the actual auto-regressive order was chosen according to the *Akaike Information Criterion* (AIC) [18].

A Kalman filter models a *linear dynamic system* (LDS) as an evolution of nonobservable variables $\vec{x}$ and observable variables $\vec{y}$, whose evolution is linked linearly through matrices $\mathbf{A}, \mathbf{B}$. The LDS, is expressed as:

$$\vec{y}_t = \mathbf{A}\vec{x}_t + V_t \quad (3)$$

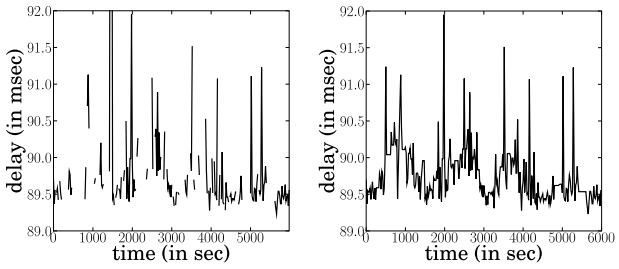
$$\vec{x}_t = \mathbf{B}\vec{x}_{t-1} + W_t \quad (4)$$

where $V_t \sim N(0, R)$ and $W_t \sim N(0, Q)$. The parameters $\mathbf{A}, \mathbf{B}, R, Q$ were estimated from the dataset we collected, using an expectation maximization (EM) algorithm for LDS [19]. Both delay and throughput are here modeled jointly, since state space representation supports multidimensional time series.

The third type of models rely on Support Vector Machines (SVMs) applied to time series prediction. The input data y used for the prediction at a given time t is a vector representing the current context: $y = [y_{t-1} \ldots y_{t-p}]'$ of the p previous measurements. By analogy with an AR model, we will refer to p as the SVM model order. Such a vector y is projected to a higher dimensional space through a non-linear mapping $\Phi(y)$. Linear modeling in such a higher dimensional space equals non-linear modeling in the original low dimensional space [17]. After defining a Lagrangian and solving for the dual variables α's, the regression function is defined as:

$$\hat{y}_t = \mathbf{f}(y) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) k(y_i, y) + b \quad (5)$$

In all our SVM experiments, the radial basis function kernel was used. This kernel, is described by $k(y_i, y) = \gamma \exp(||y_i - y||^2)$, where y_i is the input vector used to represent some past measurements in the training, y represents



(a) gapped data (b) filled data

Figure 2: Data with gaps and with gaps filled

the current context as above, and γ is a meta-parameter known as the kernel width.

3. METHODOLOGY

Data was collected between March 27$^{\text{th}}$ 2009 and April 13$^{\text{th}}$ 2009. We used five PlanetLab machines as vantage points. The machines were mostly located in Europe: two in England (London and Cambridge), one in Norway (Tromsø), one in Turkey (Istanbul), and one in USA (Pasadena).

We collected jointly delay and throughput information by monitoring (i.e., with tcpdump) a large file downloaded from FTP Linux servers. We considered 25 servers scattered around the world: Sweden, Portugal, Germany, France, Spain, Belgium, England, Luxembourg, Israel, Russia, Japan, Taiwan, USA, Brazil, and Chile.

Each vantage point downloaded the file from each FTP server every 20 minutes. The total amount of data collected was around 16GB. Five pings were made to each destination from the PlanetLab nodes. Simultaneously, an FTP file was downloaded from an FTP server, and the throughput was measured passively via tcpdump, to obtain 100 instantaneous throughput measurements. In both cases, these measurements where averaged to obtain a single data point each 20 minutes, one for delay and one for throughput.

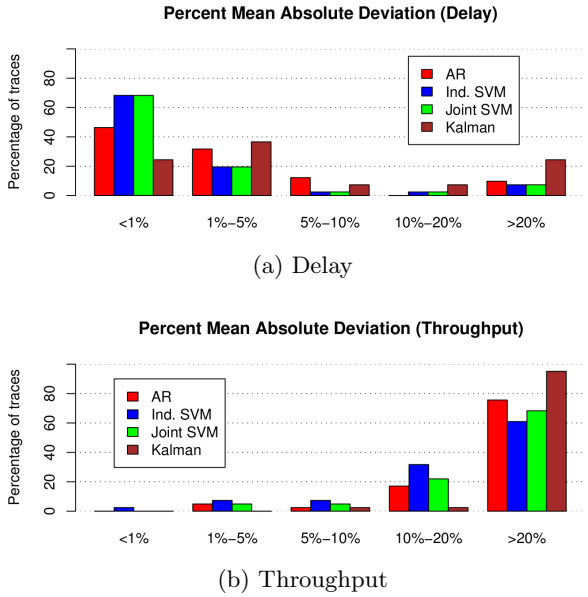
Before applying any of the methodologies described in Sec. 2, it was necessary to correct the obtained measurements. Due to network and server conditions, some of the measurements were unavailable. Indeed, if a particular server did not respond at the time of downloading the file, the measurement towards this server was given up. As a result, there were some gaps in the resulting measurements. If the amount of total missing data was superior to 35%, the trace was not used in the experiments. After filtering out this data, 41 time series, both for delay and throughput, form our actual data. Each such time series contains roughly 1480 measurements.

In order to fill the gaps, we used the *K-nearest neighbors* algorithm, with a window of six samples. The closest six neighbors, three in the past of the current missing data y_t and three towards the future of the missing data, were used. The missing data estimate $\tilde{y_t}$ is computed as the average of its neighbors:

$$\tilde{y_t} = \frac{1}{6} \sum_{k=-3}^{3} y_k \tag{6}$$

This procedure filled the gaps satisfactorily as can be seen on a typical example reported in Fig. 2.

A specific AR model was fit for each time series and each PPM. The model order selected according to AIC and averaged over the 41 time series was 8.8 for delay and 7.5 for throughput. Kalman filters jointly model delay and throughput, thus taking advantage of linear relationships between them, if any. The individual SVM approach refers to a specific SVM for each PPM time series with an input vector of 7 or 4 past measurements, respectively for delay or throughput. A joint SVM refers here to a simple extension with an additional coefficient in the input vector being the other PPM (a throughput measurement for delay prediction, or conversely) measured at the previous time step $t-1$. This extension aims at assessing whether such an additional feature may help the prediction by taking advantage of some relationships between both PPMs.



(a) Delay

(b) Throughput

Figure 3: Percentage absolute mean deviation

The initial 80% fraction of each time series (roughly 1,180 samples) was used as training data to fit the predictive models. The remaining 20% fraction (roughly 300 samples) of each time series was used for evaluating the forecasting accuracy of the predictive models. Predictive accuracy is assessed as the *Percent Mean Absolute Deviation* (PMAD), defined as:

$$PMAD = \frac{\sum_{t=1}^{N} |e_t|}{\sum_{t=1}^{N} |y_t|} \tag{7}$$

where $e_t = y_t - \hat{y_t}$ is the error, y_t is the actual value of the time series at time t and $\hat{y_t}$ is the forecasted value. The PMAD shows on average the percentual difference between the actual value and the forecasted value.

4. PRELIMINARY RESULTS

Fig. 3 reports how well on average the 20% test fraction of the 41 time series for delay or throughput are predicted. Fig. 3(a) shows very good forecasting accuracy for delay, with most of the predictions falling within the PMAD range from 0% to 5%. SVMs models are the best predictors with equivalent performances for the individual or joint version. Hence adding the measured throughput at the previous time step neither improves nor degrades delay forecasting. SVMs were used with a RBF kernel while AR models offer slightly

worse performances. This result shows the benefit of using non-linear predictors rather than linear ones. Kalman filters underperform here, hence showing that assuming a linear dependence (with Gaussian innovations) between both PPMs is not, as expected, the best modeling. Overall, the forecasting of delay is easy and individual SVMs are very good predictors.

The AR predictor is also well known to converge to the mean of the data for a long-term forecasting. The mean of the delay is actually an adequate predictor, since the variation is very small for most delay time series. However, this mean predictor must be adaptable. Fig. 1(a) shows abrupt jumps of the delay data. Most of the accuracy loss was due to predictors that were not able to adapt to a jump discontinuity in the data. This suggests that, for delay prediction, it is necessary to update the model frequently in order to detect and deal correctly with abrupt changes in the PPM value. This is an intrinsic property of the delay due to events on the network, such as switching the original route of the packets to a different one. The input vector provided to SVMs naturally adapt in such situations but the model order is fixed while it should ideally be adaptive as well.

Accurate forecasting of throughput is clearly a significantly more challenging problem. The individual SVM is again the best predictor. In contrast, the joint SVM degrades predictive performance of throughput but Kalman filters are even worse. Overall, those results call for better approaches to accurately model throughput.

5. CONCLUSION AND FUTURE WORK

Different time series prediction techniques can be used to predict PPMs, thus diminishing the need for actively probing the network. Depending on the PPMs under study, the problem can be more or less difficult. Delay can be predicted with very good accuracy, in particular with support vector machines (SVMs). However, the model must be constantly adapted in order to deal with jump discontinuities. Our future work includes an explicit modeling of those jumps. For instance, some simple statistics can be computed from the observed lengths between consecutive jumps in the training data. The probability of a jump is expected to increase with the time interval since the last observed jumps. A rough estimate of such probability could serve, for instance, as an additional input feature to SVM models.

Throughput is much more difficult to predict due to its strongly nonlinear behavior. In this preliminary work, a very standard RBF kernel was used within SVM models. Improvements could be obtained by considering dedicated kernels for time series such as those proposed by Rüping [20]. Alternative representations of these time series will also be further studied.

It is expected that there are statistical relationships between delay and throughput, although possibly fairly complex ones. Our preliminary results show that measuring throughput at the previous time step does not help the delay forecasting, and conversely. Additional experiments (not detailed here) have shown that increasing the number of time steps for which throughput is measured before predicting delay is even worse. Those results call for more sophisticated ways to model dependencies between PPMs. It is worth noting than Kalman Filters form a special case of Gaussian Processes (GPs) [21]. GPs, being kernel methods, can look for dependencies modeled through a non-linear mapping of the data to a higher dimensional space. This could offer an interesting alternative to the simplistic assumption of linear dependencies in Kalman filters.

6. REFERENCES

[1] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," IETF, RFC 1633, June 1994.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF, RFC 2475, December 1998.

[3] S. Agarwal, C.-N. Chuah, and R. H. Katz, "OPCA: Robust interdomain policy routing and traffic control," in *Proc. IEEE Conference on Open Architectures and Network Programming (OPENARCH)*, April 2003.

[4] A. Akella, B. Maggs, S. Sheshan, A. Shaikh, and R. Sitaraman, "A measurement-based analysis of multihoming," in *Proc. ACM SIGCOMM*, August 2003.

[5] N. Feamster, Z. M. Mao, and J. B. Rexford, "Borderguard: Detecting cold potatoes from peers," in *Proc. ACM SIGCOMM IMC*, Oct. 2004.

[6] European Commission, "European fire initiative," http://cordis.europea.eu/fp7/ict/fire/.

[7] "Global environment for network innovations," http://www.geni.net.

[8] A. Bavier, N. Feamster, M. Huang, J. Rexford, and L. Peterson, "In VINI veritas: Realistic and controlled network experimentation," in *Proc. ACM SIGCOMM*, Sep. 2006.

[9] D. Saucez, B. Donnet, and O. Bonaventure, "On the impact of clustering on measurement reduction," in *Proc. IFIP Networking*, May 2009.

[10] M. Crovella and B. Krishanmurthy, *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley and Sons, 2006.

[11] L.-J. Chen, C.-F. Chou, and B.-C. Wang, "A machine learning-based approach for estimating available bandwidth," in *Proc. IEEE TENCON*, October 2007.

[12] M. Yang, J. Ru, X. R. Li, H. Chen, and A. Bashi, "Predicting Internet end-to-end delay: A multiple-model approach," in *Proc. IEEE INFOCOM*, April 2006.

[13] R. Beverly, K. Sollins, and A. Berger, "SVM learning of IP address structure for latency prediction," in *Proc. ACM SIGCOMM Mining Network Data Workshop (MINENET)*, September 2006.

[14] Y. Zhang and N. Duffield, "On the constantcy of Internet path properties," in *Proc. ACM Workshop on Internet Measurement (IMW)*, October 2001.

[15] G. Box and G. Jenkins, *Time series analysis, forecasting and control*. Holden-Day, Incorporated, 1990.

[16] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[17] K.-R. Müller, A. J. Smola, R. Rötsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, "Predicting time series with support vector machines," in *Proc. 7th International Conference on Artificial Neural Networks (ICANN)*, October 1997.

[18] C. Hurvich and C. Tsai, "Regression and time series model selection in small samples," *Biometrika*, vol. 76, no. 2, pp. 297–307, 1989.

[19] Z. Ghahramani and G. E. Hinton, "Parameter estimation for linear dynamical systems," University of Toronto, Department of Computer Science, CRG-TR 92-2, February 1996.

[20] S. Rüping, "Svm kernels for time series," in *Proceedings LLWA*, 2001.

[21] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.