Predictive Estimation of Wireless Link Performance from Medium Physical Parameters Using Support Vector Regression and k-Nearest Neighbors

Guillaume Kremer, Philippe Owezarski, Pascal Berthou, German Capdehourat

Predictive Estimation of Wireless Link Performance from Medium Physical Parameters Using Support Vector Regression and *k*-Nearest Neighbors

Guillaume Kremer[1,2], Phillippe Owezarski[1,2], Pascal Berthou[1,2] and German Capdehourat[3]

[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, UPS, INSA, LAAS, F-31400 Toulouse, France
[3]Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Uruguay
Email: {kremer, owe, berthou}@laas.fr, gcapde@fing.edu.uy

Abstract. In wireless networks, the physical medium is the cause of most of the errors and performance drops. Thus, an efficient predictive estimation of wireless networks performance w.r.t. medium status by the communication peers would be a leap ahead in the improvement of wireless communication. For that purpose, we designed a measurement bench that allows us to accurately control the noise level on an unidirectional WIFI communication link in the protected environment of an anechoic room. This way, we generated different medium conditions and collected several measurements for various PHY layer parameters on that link. Using the collected data, we analyzed the ability to predictively estimate the throughput performance of a noisy wireless link from measured physical medium parameters, using machine learning (ML) algorithms. For this purpose, we chose two different classes of ML algorithms, namely SVR (Support Vector Regression) [1] and k-NN (k-Nearest Neighbors) [2], to study the tradoff between complexity and estimation accuracy. Finally, we ranked the pertinence of the most common physical parameters for estimating or predicting the throughput that can be expected by users on top of the IP layer over a WIFI link.

1 Introduction

Wireless networks are of essential importance nowadays. Users are more and more mobile and access the Internet thanks to mobile devices as laptops, smart phones or tablets. Even when staying at home, users want to get rid of wires. However, the wireless medium does not provide the same capabilities as wired networks on copper or fiber. In wireless networks, the physical medium is limited in terms of capacity, and the cause of most of the errors and performance drops. From a user or administrator point of view, the quality of wireless communication can appear as very versatile and unpredictable. This makes wireless

networks very complex to manage, and users often experience communication quality drops that are completely unexpected.

Monitoring wireless networks is then very difficult. Monitoring such networks at the IP layer is very inefficient (whereas it is the way it is done in wired networks with extremely good results). Some previous work tried to include the MAC level in the monitoring of wireless networks [3], but none integrates the full monitoring of the network from physical to network layers. We nevertheless argue that this is the direction to follow, and propose our preliminary study to estimate the relations between the physical signal parameters and the performance at the network level. Physicists are doing very strong studies on the signal level, but do not study the impact on upper layers [4]. In this paper, it is proposed to bridge the gap between the signal and the digital world in wireless communication networks.

This paper then presents a double contribution.

First, we designed and built a platform for benchmarking wireless communications. Many wireless testbeds, identified in the literature, already exist for that purpose. However, the major trend is to build large grid of wireless nodes which can be programmed individually to transmit, receive and/or measure data. Custom topologies can be made out of the grid by switching on and off nodes. For example, Orbits [5] follows this approach. However, these platforms are built in open environments and lack the isolation and environmental control required to conduct an accurate cross-layer study on wireless networks. Contrary to these works, our testbed is built in an anechoic chamber to fully control the experimental environment, and avoid external signals to disturb the behavior of the communicating devices and the quality of the measurements. We used on this platform the common digital communications devices that are widely used (laptops, tablets, smart phones), as well as dedicated signal measurement tools specifically designed for physicists. Anyway, because of space limit, this paper concentrates on the study of a WIFI link.

Second, the paper presents the analysis of the relations between the PHY parameters of the WIFI connection, and the performance parameters on top of the IP layer. It aims at demonstrating that, at the opposite of wired networks, the monitoring of wireless network can not avoid monitoring the physical level. It is shown that using a very limited number of signal parameters (one or two), it is possible to very accurately estimate communication performance and quality parameters as network level throughput, delay or loss ratio. With a carefully selected and set ML algorithm, it is even possible to predict performance drops at the scale of one second. For this purpose we rely on two kinds of supervised ML algorithms: SVR and k-NN. Both of them are known to have good prediction capabilities and to succeed in many domains as long as these domains can provide accurate time series [2, 6]. However their operational characteristics are very different making them more prone to different usage and applications. For example, SVR algorithms are strong learners whereas k-NN's learning is weak, thus making them unable to assimilate training data on the fly because of the huge computational complexity. However, SVR algorithms

are more sophisticated than k-NN and so are more efficient to generalize data and usually more accurate on the estimations [2]. Therefore, we will compare the relative estimation performances obtained with SVR and k-NN as well as their performance concerning their time of execution (learning and estimation delays). Again, because of space limit, the paper only presents the results with the most common physical signal parameters as SNR or RSS for estimating the throughput obtained on top of the IP layer.

2 Machine learning algorithms

2.1 SVR theory

This section presents the basic theory behind SVR. More details can be found in [7]. Given a set of training data $\{(x_1, y_1), ..., (x_n, y_n)\} \in \mathbb{X} \times \mathbb{R}$ with $\mathbb{X}$ the input space. The purpose of SVR algorithm is to estimate a function $f(x)$ with the requirements of having at most ϵ deviations from the targets y_i. Equations (1) and (2) show respectively SVR approximation for linear and non-linear form, with $\langle ., . \rangle$ the notation for the dot product in $\mathbb{X}$. In the linear case, SVR performs a linear regression in the input space. In the non-linear case, no regression can be done in the input space. Therefore, on a first hand, the SVR algorithm has to map the data into some feature space $\mathbb{F}$ via the function $\phi : \mathbb{X} \to \mathbb{F}$. On a second hand, the classical SV regression algorithm is applied in the new feature space.

$$f(x) = \langle w, x \rangle + b \text{ with } w \in \mathbb{X} \text{ and } b \in \mathbb{R}. \tag{1}$$

$$f(x) = \langle w, \phi(x) \rangle + b \text{ with } w \in \mathbb{X} \text{ and } b \in \mathbb{R}. \tag{2}$$

The second requirement for the regression is to maximize the "flatness" of the weights, here measured by $\|w\|^2$. Hence, in the non-linear case both coefficients w and b are estimated by minimizing the regularized risk function given in (4). In this equation, C is a user-defined constant which controls the trade-off between the training error and the model flatness. L_ϵ is the ϵ-insensitive loss function defined by equation (3). This function allows the SVR algorithm to only penalize estimation errors greater than ϵ.

$$L_\epsilon(y_i, f(x(i), w) = \begin{cases} |y_i - f(x(i), w)| - \epsilon & \text{if } |y_i - f(x(i), w)| \geq \epsilon. \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$$R(f, C) = C \sum_{i=1}^{n} L_\epsilon(y_i, f(x(i), w)) + \frac{1}{2} \|w\|^2 . \tag{4}$$

To complete the regression we need to solve a convex optimization problem, which is more easily done by maximizing its dual form and introducing the

Lagrange multipliers (α_i,α_j^*). The new optimization problem is given by (5) and is subject to $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$ and $\alpha_i^* \in [0, C]$.

$$\begin{array}{ll} \text{Maximize} & -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle\phi(x_i, x_j)\rangle \\ & -\epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y(i)(\alpha_i - \alpha_i*). \end{array} \tag{5}$$

Solving this leads to a new definition of (2) as $f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\langle\phi(x_i), \phi(x)\rangle + b$.

At this point, this definition shows that the solution can be found by only knowing $\langle\phi(x_i), \phi(x)\rangle$ instead of explicitly knowing ϕ. A function $k(x, x')$ which corresponds to a dot product in some feature space $\mathbb{F}$ as defined by $k(x, x') = \langle\phi(x), \phi(x')\rangle$ is called a kernel. This kernel function can be any symmetric function satisfying Mercer condition such as the Gaussian Radial Basis (RBF) which is defined by $K(x_i, x_j) = \exp(-\gamma \left\|x_i - x_j\right\|^2)$. The Gaussian kernel is parametrized by γ ($\gamma > 0$) which impacts the generalization capability of the regressor among other things.

2.2 *k*-NN for continuous variables estimation theory

The learning approach of k-NN [8] is to memorize the entire training set. As so, the algorithm belongs to the class of the so-called lazy learners as [9, 10] for instance. Given a set of training data $D = \{(x_1, y_1), ..., (x_n, y_n)\} \in \mathbb{X} \times \mathbb{R}$, with $\mathbb{X} \subseteq \mathbb{R}$, the process followed by k-NN to estimate an object $z = (x', y')$ can be easily summed-up in three steps. Firstly, the algorithm computes the distance $d(x', x)$ between z and every object $(x_i, y_i) \in D$. Secondly, the set F of the k closest neighbors to z is selected. Thirdly, k-NN computes the estimation as $\hat{y} = \frac{1}{k}\sum_{i=1}^{k} x_i$ with $x \in F$. Variants exist and concern essentially the method used to compute the distance $d(x, x')$ such as the Manhattan, Euclidean or Minkowski distance. The p-order Minkowski distance for two sets of points $F = (x_1, ...x_n)$ and $G = (y_1, ..., y_n) \in \mathbb{R}^n$ is defined by $(\sum_{i=1}^{n} |x_i - y_i|)^{\frac{1}{p}}$.

3 Experimental platform and dataset

3.1 Experimental conditions and measurement equipments

The implementation of a dedicated wireless testbed is a major requirement for our work. First of all, experimentations must be reproducible, allowing comparison between different sets of measurements and algorithms. This point is not trivial when using wireless networks as the environment factors have a high impact on the network performances. Secondly, part of the originality of this work comes from the combination of measurements made at multiple network layers, using electronics instruments and software tools. This was also a strong requirement to be able to monitor the physical layer (the wireless transmission), and compare it to the higher layers, from the mac layer information given by the network cards to the end-to-end layers as transport throughput for instance. The

hardware introspection requirement has an impact on the components choice as explained below. Thirdly, the synchronization of all of these datasets was a sticky point, but absolutely required to ensure a good behavior of the learning algorithms.

3.2 Reproducibility requirement

Our wireless testbed was designed inside an anechoic room. An anechoic room is a protected RF room which simulates free space conditions. Our model of chamber is 4,10 meters long for 2,50 meters wide. Inside, walls are covered of microwave absorbers materials that break and scatter any wireless signal that would come from an inside source. The chamber is then free of any multi-path propagation. There are different types of absorbers, each of them is defined for a specific frequency range that allows us to use the anechoic chamber for different purposes and frequencies. The absorbers protect also the inner environment of the room from outside perturbations. This protected context minimizes the uncontrolled parameters of our communication.

3.3 Introspection requirement and components choice

Inside the anechoic chamber we placed two WIFI nodes. The nodes are controlled through a wired network to avoid interference with the wireless communication. The nodes are Avila-GW2348-4 gateway platforms and run a Linux OpenWrt OS. The boxes have an Intel Xscale processor, 64 MB of SDRAM and 16MBytes of Flash memory. The WIFI network controllers are based on the AR5414 chipset from Atheros which uses the ath5k driver and are attached to an omni-directional antenna. The choice of the wifi chipset and its driver was crucial because they define the amount of metrics and the accuracy that it will be possible to obtain. The ath5k driver is open-source and well documented thanks to an active online community support. It has also a good integration within the OpenWrt OS. The OpenWrt OS is flexible enough to allow the implementation of new functionalities so that it accelerates the upgrade of the bench. In addition and because we were unable to capture the noise strength of the received signal with the Atheros hardware, we used an oscilloscope connected to the receiver antenna. It records the amplitude of the received signal. The oscilloscope chosen was a fast Lecroy WaveRunner which allows us to capture a maximum number of frame signal with little loss and to record them on internal memory. The precision of this instrument gives us the ground truth required by the training methods used. It also embeds a large library of filters, and operators which can be applied on the input signals. The oscilloscope is also synchronized by NTP.

Synchronization requirement As we used several equipments to get measurements, it is needed to have their clock very accurately synchronized. This was done with NTP by using a dedicated wired connection to a remote NTP server (accuracy with a shared network bus is not sufficient).

Capture and measurement processes The configuration of the network interfaces is done in promiscuous mode to capture any packets sensed by their antenna. The packets are captured at the MAC layer using the PCAP library and tools when they arrive at the kernel interface. The packets contain data from link to application layers, such as the 802.11 channel number, the type of frame at the MAC layer, or packet size at the network layer. Additionally, a packet also contains a RADIOTAP header which gives radio level information such as the received signal strength (RSS) reported by the ath5k driver. We modified the ath5k drivers of the OpenWrt OS to permit, when possible, the propagation of packets with frame check sequence (FCS) errors to the upper layers, while on the original kernel they were discarded. The propagation is only possible if the error corrupted the data but not the header fields. Following this modification the RADIOTAP header now contains a flag specifying whether a FCS error was detected when decoding the packet.

The Lecroy oscilloscope was set to capture and flush the data as soon as a frame is detected on the input cable. This happens when the amplitude of the sensed signal is above a specific threshold, set to be in between the current noise floor and the minimal amplitude value of a frame. This threshold has to be set in a way to prevent exceptional high noise values that could be incorrectly detected as a frame.

3.4 Experimental protocol

Noise generation. One of the objectives of our environment is to minimize the presence of these uncontrolled parameters on the communication. Another objective is to generate and control selected parameters that will impact our communications.

The noise and the interferences significantly impact the communication. We then inject noise in the environment using a signal generator to perturb the communication. The signal generator is a device which emits RF signals. It can be configured to generate very realistic noise. Among the parameters of the generated noise, two important elements have a crucial impact: on a first hand the modulation used characterizes the main characteristics of the noise signal in the time and frequency domains (i.e. it characterizes the spectral occupancy of the generated signal, its fading or narrowness). On a second hand, the amplitude of the signal also affects the measured level of noise on the receiver side. We found that the AWGN (Adaptive White Gaussian Noise) noise modulation was a good choice for our preliminary studies because of its simplicity. Moreover it can be used to impact the entire bandwidth of a 802.11g channel contrary to most other modulation schemes which produce narrow band noise. The noise level was determined empirically by testing the effects on the communication. Finally, a major element that affects the noise generated in the anechoic chamber is the antenna. It characterizes the waveform, the direction and the amplitude of the noise wave. In order to perturb only one side of the communication we used a very directional antenna pointed to the receiving station. We use IPERF to

Table 1: Constitutions and characteristics of our training sets. Each vector represents 1 second of measurements

Training set	**Dataset definition**
notation	{Tx Power (dBm); Noise Power (dBm)}; {sample 2};...
*Dataset*1 (5323 vectors)	{10;-20};{10;-17};{10;-15};{10;-13};{10;-10};{10;-7};{10;-5}; {20;-20};{20;-17};{20;-15};{20;-13};{20;-10};{20;-7};{20;-5}
*Dataset*2 (2661 vectors)	{10;-20};{10;-17};{10;-15};{10;-13};{20;-20};{20;-17};{20;-15};{20;-13}
*Dataset*3 (1330 vectors)	{10;-20};{10;-17};{10;-15};{10;-7};{10;-5};{20;-20};

generate traffic between the two peers. The traffic is a TCP flow with a constant throughput of 24 Mb/s. The size of the packets is set to 1470 bytes.

Training and datasets We generated different samples with different noise levels and different transmission powers. All the samples have the same duration of 5 minutes and will be used to constitute our training datasets. Table 1 sums up the characteristics of the different samples. The same experimental settings (transmission power and noise) are used for training and testing. Therefore a training dataset which contains all these samples will be considered as having full knowledge about the possible use cases met in the test dataset. Hence, to test the generalization capacity of our algorithm, we built three different training datasets as described in table 1. These datasets differ by the quantities of samples they are made of, and consequently by the level of knowledge they represent.

3.5 SVR features definitions

Atheros Received Throughput This is the performance metric of the communication that we are considering in this paper. It is computed from the PCAP captured at the receiver side of the transmission. It is defined by $BW_i = \sum_{k=1}^{n} L(p_k)$ with $k \in \mathbb{N}$. BW_i is the computed throughput at second i, $L(p_k)$ is the length of the payload at the network layer for packet p_k such as $p_k \in P_i$ which is defined as the set of the n^{th} received packets without FCS error during second i: $P_i = \{p_1, ..., p_n\}$.

Atheros RSS The Atheros RSS is extracted from the RSS field in the RADIOTAP headers of the packets included in the PCAP files. Given that $RSS(p_k)$ is the RSS of packet p_k such as $p_k \in P_i$, and R_i is the set of RSS extracted from packets captured during second i, it is defined as $ATH\_RSS_i = \overline{R_i}$ with $R_i = \{RSS(p_1), ..., RSS(p_n))\}$.

Lecroy noise In addition to the Atheros values, we extract different metrics from the Lecroy datasets. These values are computed from the Root Mean Square (RMS) values of the raw data. These RMS values can be split into three parts, which are the data that are before, during and after the frame. The part of the data before and after the frame are the noise values and therefore can be used to extract the noise floor during the reception of that frame. We consider A and C, the sets of these points. Therefore we compute the average noise floor of the data during the reception of frame f with $N_f = \overline{A \cup C}$.

With M_i the set of noise levels extracted from the frames captured by the Lecroy oscilloscope during second i, we compute the feature for the noise floor at second i $LECR\_NOISE_i$ as $LECR\_NOISE_i = \overline{M_i}$ with $M_i = \{N_{p_1}, ..., N_{p_n}\}$ and $p_k \in P_i$.

Lecroy RSS The RSS of the received frame is computed on the first 8 symbols to comply with 802.11 standard (see *http://standards.ieee.org/getieee802/*). These points constitute the set D. Thus, similarly to previous equations, the RSS for a frame f is given by $R_f = \overline{D}$ and $LECR\_RSS_i = \{R_{p_1}, ..., R_{p_n}\}$, where $LECR\_RSS_i$ is the feature of the Lecroy RSS at second i.

Lecroy SNR Finally we compute the SNR S_f for frame f as the difference between the noise floor and the RSS of the frame P and therefore, similarly to previous formulas: $S_f = R_f - N_f$ and $LECR\_SNR_i = \overline{W_i}$ with $W_i = \{S_{p_1}, ..., S_{p_n}\}$ and $p_k \in P_i$.

4 Estimation of the relations between physical and performance parameters in WIFI communications

4.1 ML based methodology

The 2nd contribution of this paper is the analysis of the relations linking the PHY layer parameters and the upper layers performance.

SVR SVR algorithm has been used with RBF as a kernel function. As section 2.1 points it out, in our configuration SVR requires three user-defined parameters (C, γ and ϵ) which can impact performance and therefore must be carefully selected with regard to the application. For our estimations, we used a grid search to select these SVR parameters. It is a common empirical method which consists in an exhaustive test run of SVR training using generated settings combinations. We then select the best combination of C, γ and ϵ among the results.

***k*-NN** For the performance of k-NN, the value of k must be carefully selected. Therefore, after several tests on the different datasets, we chose a value which allows a good tradeoff between the estimation accuracy and the generalization

results. Hence, in the presented experimentation, we set the value of k to 3. The distance method used is Minkowski with order 2 which corresponds to the Euclidean distance recommended with the traditional version of the algorithm [8].

Training and estimation delays measurements One part of the analysis of the machine learning estimations concerns the computational time associated with the training and estimations process. Our ML setup uses Python scikit-learn implementation [11] of SVR and k-NN. The delays are computed by reading the current clock using the 'time' function. The clock is read twice: before and after the measured process. The difference of the two measures constitutes the delay for the measured process. For each estimation, we made 100 runs and then computed the average and standard deviation of the delays. The CPU used to conduct the measures is a 64 bits Intel Core 2 Duo (2x2.53 GHz) with 6 MB of cache memory. The computer disposes of 4 GB of RAM memory. The operating system is Debian Linux.

4.2 Estimation performance

To evaluate the estimations, two methods are used.

Mean Squared Error (MSE) Given that $\hat{Y_i}, ..., \hat{Y_n}$ are estimations and $Y_i, ..., Y_n$ are the real values, the MSE is defined as $MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y_i} - Y_i)^2$.

Percentage of correct estimations We also use the percentage of correct estimations noted $P(e < d)$ and defined by $P(e < d) = \frac{1}{n}\sum_{i=1}^{n} D(\hat{Y_i}, Y_i, d)$. This value is the percentage of estimations which differ from the corresponding real values by less than a defined threshold d as shown on equation (6). These estimations are then considered 'correct'. Given the maximum throughput of 24 Mbps and the size of the packets defined to be 1470 bytes, we set the value of the threshold d to 1 Mbps. Indeed, this threshold corresponds to an error in the estimation of 4% (89 packets over 2139 transmitted during one second). By considering the preliminary measured performance of the algorithms this value could be considered to be fair to assess the goodness of the algorithms.

$$D(\hat{Y_i}, Y_i, d) = \begin{cases} 1 & \text{if } |\hat{Y_i} - Y_i| < d. \\ 0 & \text{if } |\hat{Y_i} - Y_i| \geq d. \end{cases} \tag{6}$$

Table 2: Results of the estimations using physical layer metrics. $D1$, $D2$ and $D3$ stands respectively for $Dataset1$, $Dataset2$ and $Dataset3$.

(a) Scores of the estimations.

n°	Physical layer parameter(s)	MSE (Mbps^2)						P($e < 1Mbps$) (%)					
		SVR			k-NN			SVR			k-NN		
		$D1$	$D2$	$D3$	$D1$	$D2$	$D3$	$D1$	$D2$	$D3$	$D1$	$D2$	$D3$
1	ATH_RSS	11.24	11	10.17	23	33	34	35	33	34	24	22	14
2	$LECR\_RSS$	4.42	3.9	4.5	27	7.1	10	51	59	32	18	35	31
3	$LECR\_NOISE$	2.28	5.4	5.8	5	2.8	4.2	69	55	44	50	44	24
4	$LECR\_SNR$	1.69	**1.6**	**1.6**	4	2.3	2.8	64	**66**	**62**	48	50	45
5	$ATH\_RSS + LECR\_NOISE$	1.02	2.3	3.3	4	1.3	**1.7**	70	49	41	54	60	**50**
6	$LECR\_RSS + LECR\_NOISE$	**0.88**	2.0	2.53	**2**	**1.2**	2.2	**75**	57	49	**64**	**63**	46

(b) Pertinence of the estimations.

n°	Physical layer parameter(s)	SVR Pertinence ranking						k-NN Pertinence ranking					
		MSE			P($e < 1Mbps$)			MSE			P($e < 1Mbps$)		
		$D1$	$D2$	$D3$	$D1$	$D2$	$D3$	$D1$	$D2$	$D3$	$D1$	$D2$	$D3$
1	ATH_RSS	6	6	6	6	6	5	6	6	6	5	6	6
2	$LECR\_RSS$	5	4	4	5	2	6	5	5	5	6	5	4
3	$LECR\_NOISE$	4	5	5	3	4	3	4	4	4	3	4	5
4	$LECR\_SNR$	3	**1**	**1**	4	**1**	**1**	2	3	3	4	3	3
5	$ATH\_RSS + LECR\_NOISE$	2	3	3	2	5	4	2	2	**1**	2	2	**1**
6	$LECR\_RSS + LECR\_NOISE$	**1**	2	2	**1**	3	2	**1**	**1**	2	**1**	**1**	2

4.3 Estimation results

Table 2 contains the results of the throughput estimation based on 6 different PHY or combinations of PHY parameters for respectively$Dataset1$, $Dataset2$, and $Dataset3$. The first column quotes the PHY parameters that have been used for the SVR estimation of the IP throughput. Columns 2 to 4 show the figures obtained for the MSE and the probability $P(e < 1Mb)$ for both ML algorithms. The four last columns give the ranking for the PHY parameters according to their ability to allow good estimations of the throughput. A ranking of 1 corresponds to the best result among the 6 PHY parameters considered.

For $Dataset1$, i.e. the full one, the best result is obtained with $LECR\_RSS+$ $LECR\_NOISE$ for both families of algorithms. The estimations for SVR are plotted on figure 1. This figure exhibits impressive matching between the real and estimated values of the throughput, with just very few outliers appearing (75% matchings). We got as impressive results for $Dataset2$, and $Dataset3$, but this time, the best results for SVR have been obtained with the $LECR\_SNR$ parameter (60% matchings). The difference of the results when using a full trace for the training compared to a sampled one exhibits the non empty intersection between PHY parameters as SNR, RSS and NOISE. These 3 parameters are

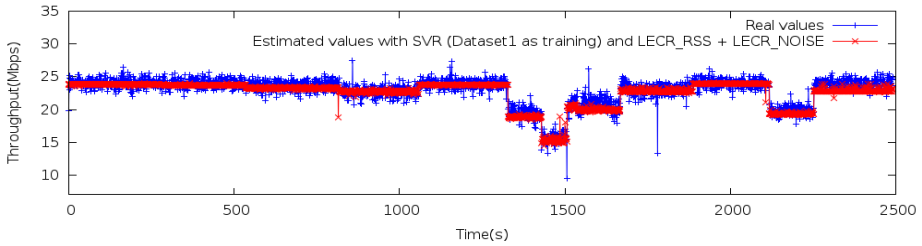

Fig. 1: Throughput estimation results obtained with the $LECR\_RSS + LECR\_NOISE$ metric compared to the real throughput.

closely related. The results for k-NN improve with the use of $Dataset2$. Contrary to SVR, the best estimations are obtained with the features 5 and 6 for every training datasets. Generally speaking, SVR performs better than k-NN excepts in the 2nd training dataset where k-NN outperforms SVR in terms of MSE. It nevertheless clearly appears with these figures that SNR, RSS and NOISE can help to perfectly estimate and predict (on a one second scale) the performance of the network at layers 3 and 4. Nevertheless, a deeper analysis on larger datasets, that still need to be produced, would allow a more accurate characterization of the link between PHY parameters and network performance. Actually, it appears that while the combined features metrics performance decreases, the overall performance of the RSS metrics 1 and 2 increases or stays more or less the same. This seems to suggest that the full training set was not adapted to these metrics. This is even more visible in k-NN results, while MSE performances improve impressively between $Dataset1$ and $Dataset2$.

The difference between the full and the reduced sets is that the samples obtained with high noise are not present in the reduced datasets. This could be caused by incoherent values existing in $Dataset1$ because of the bad and noisy conditions. One possibility is that these values could deteriorate the model issued from the training process. This hypothesis seems to be corroborated by the results obtained with k-NN and the simplicity of its algorithm which makes it more sensible to the general quality of the training dataset and the choice of the feature. This aspect needs to be considered for improving our platform and experiment protocol.

4.4 Training and estimation time performance

Table 3 presents the results of the measured delays for training and estimations using SVR and k-NN. According to these numbers, the time taken by SVR to train can be very high. Hence, with $Dataset1$ and the RSS metrics, the delays goes up to the tens of seconds. Then the time decreases with the use of smaller training sets. In the case of k-NN, no model are computed, the data are simply memorized. Therefore the training is very fast and essentially depends on the size of the training sets. As a consequence, k-NN values decrease geometrically by a

Table 3: Results of the measured delays for training and estimations using physical layer metrics. $D1$, $D2$ and $D3$ stands respectively for $Dataset1$, $Dataset2$ and $Dataset3$.

(a) Average delays observed for the training processes on 100 runs (values into brackets are the standard deviation of the distributions. Due to space limitation, standard deviation values are given in 10^3 unit).

n°	Physical layer parameter(s)	Time used for training (s)					
		SVR			k-NN		
		$D1$	$D2$	$D3$	$D1$	$D2$	$D3$
1	ATH_RSS	5.39 *(40)*	1.40 *(2)*	0.36 *(0.4)*	0.048 *(2)*	**0.023** *(0.1)*	**0.012** *(0.1)*
2	$LECR\_RSS$	41.27 *(300)*	11.71 *(7)*	3.12 *(2)*	0.048 *(1)*	**0.023** *(0.2)*	**0.012** *(0.1)*
3	$LECR\_NOISE$	5.17 *(10)*	1.38 *(1)*	0.36 *(0.8)*	0.051 *(6)*	**0.023** *(0.2)*	**0.012** *(0.1)*
4	$LECR\_SNR$	11.54 *(6)*	3.87 *(4)*	1.35 *(2)*	0.048 *(4)*	**0.023** *(0.2)*	**0.012** *(0.1)*
5	$ATH\_RSS + LECR\_NOISE$	**4.50** *(4)*	**1.15** *(2)*	0.30 *(0.2)*	0.048 *(0.6)*	**0.023** *(0.1)*	**0.012** *(0.1)*
6	$LECR\_RSS + LECR\_NOISE$	4.72 *(9)*	1.23 *(2)*	**0.31** *(3)*	**0.046** *(0.5)*	**0.023** *(0.1)*	**0.012** *(0.08)*

(b) Average delays observed for the estimations processes on 100 runs (values into brackets are the standard deviation of the distributions. Due to space limitation, standard deviation values are given in 10^3 unit).

n°	Physical layer parameter(s)	Time used for estimation (s)					
		SVR			k-NN		
		$D1$	$D2$	$D3$	$D1$	$D2$	$D3$
1	ATH_RSS	1.52 *(10)*	0.78 *(6)*	0.40 *(3)*	1.13 *(10)*	0.63 *(1)*	0.39 *(0.6)*
2	$LECR\_RSS$	1.58 *(20)*	0.81 *(10)*	0.42 *(3)*	0.61 *(3)*	0.44 *(0.8)*	0.29 *(0.6)*
3	$LECR\_NOISE$	1.47 *(20)*	0.76 *(10)*	0.42 *(4)*	0.96 *(30)*	0.35 *(0.9)*	0.08 *(0.3)*
4	$LECR\_SNR$	1.38 *(30)*	0.70 *(10)*	0.36 *(3)*	0.74 *(60)*	0.45 *(0.8)*	0.19 *(0.3)*
5	$ATH\_RSS + LECR\_NOISE$	**1.34** *(9)*	**0.67** *(10)*	**0.35** *(8)*	0.32 *(10)*	**0.15** *(0.3)*	**0.06** *(0.1)*
6	$LECR\_RSS + LECR\_NOISE$	1.38 *(1)*	0.70 *(4)*	0.36 *(3)*	**0.27** *(0.4)*	**0.15** *(0.2)*	0.08 *(0.1)*

factor of 2 when changing from $Dataset1$ to $Dataset2$ and then from $Dataset2$ to $Dataset3$. According to section 2.1, SVR forces the estimated function to be within an ϵ distance of the averaged data, a requirement which can be tedious for the algorithm to fulfill. Hence, the high value for SVR model training are explained by the usage of this ϵ parameter which affects greatly the training accuracy as well as the delays. However, this affirmation would need more study focused on the SVR parameters and these specific data. The time taken for the estimation are higher when using SVR, than when using k-NN. The SNR delays vary with the size of the training set. This result seems unintuitive since SVR training model is based on regression. However, the results obtained with k-NN are conform to its training model which is based on the memorization of the entire training set. k-NN results are very good comparatively to the one of SVR.

By observing the global results, we see that k-NN can largely compete with SVR when it comes to accuracy while at the same time being slightly faster.

5 Conclusions and future work

The main contribution presented in this paper deals with the design of a generic platform for monitoring and analyzing wireless networks. This wireless testbed is set in the RF protected environment of an anechoic room, allowing us to control the perturbation on the physical medium by generating noise. It also has the originality to integrate pure physical signal measurement tools as Lecroy oscilloscopes for very accurate measurements serving as ground truth. Based on the collected data, the second contribution of the paper deals with exhibiting the importance of PHY parameters on network communication performance. The correlation between the physical environment and the communication performance is so strong that it is possible by only monitoring the SNR and the RSS of the signal to predict the performance level at the TCP/IP level. This result has been demonstrated using different kinds of models, in particular the SVR and k-NN models presented in this paper. Future work includes a large exploitation of our platform. Indeed, for this preliminary stage, we just set simple scenarios with a single connection and simple noise model that can appear a bit far from realistic situations. These first simplistic scenarios were manadatory to validate the platform accuracy, and the monitoring and analysis tools, as well as for gaining the required skills required for this multi-thematic work, especially in the domain of the signal propagation and behaviour. We now plan to generate large datasets with more complex and realistic scenarios, and this for different kinds of wireless networks, including WIFI, UMTS, LTE, etc. We will also exploit this datasets by deeply analyzing them, understand how wireless networks behave, and then trying to improve the way we use and manage them.

6 Acknowledgments

This work is partially funded by the French National Research Agency (ANR) under two projects: the MAITRE project of the STIC AmSud program, and the RESCUE project of the VERSO program.

References

1. V. Vapnik, S. E. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems 9*, vol. 9, 1997.
2. X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, Dec. 2007.

3. T. Claveirole and M. D. de Amorim, "Wipal and wscout, two hands-on tools for wireless packet traces manipulation and visualization," in *ACM Mobicom Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, 2008.
4. A. Lecointre, D. Dragomirescu, and R. Plana, "New methodology to design advanced mb-iruwb communication system," *IEE Electronics Letters*, vol. 11, 2008.
5. D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 3, 2005, pp. 1664–1669 Vol. 3.
6. N. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *Computational Intelligence Magazine, IEEE*, vol. 4, no. 2, pp. 24–38, 2009.
7. A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, Aug. 2004.
8. N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. pp. 175–185, 1992. [Online]. Available: http://www.jstor.org/stable/2685209
9. C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artif. Intell. Rev.*, vol. 11, no. 1-5, pp. 11–73, Feb. 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1006559212014
10. A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Commun.*, vol. 7, no. 1, pp. 39–59, Mar. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=196108.196115
11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.