RESCUE D2.3: Integration of monitoring and metrology tools in RESCUE

Guillaume Kremer, Denis Carvin, Philippe Owezarski, Pascal Berthou

Title	Integration of monitoring and metrology tools in RESCUE
Summary	This is the final report of WP2 that reports on the integration of monitoring research lead in RESCUE.
Responsible	LAAS
Contributors	Guillaume Kremer, Denis Carvin, Philippe Owezarski, Pascal Berthou
Linked to WP	2

1 Deliverable goal and content

The idea behind the monitoring in the RESCUE project deals with monitoring wireless networks from the signal at the air interface up to the digital data. The expectation is that by monitoring the signal at the physical level, it would be possible to infer the behavior of these networks at all upper layers, and to predict the evolution of the physical medium and of the services provided on the upper layers, at least on a few hundreds of milliseconds. Based on this information, it would then be possible to anticipate the need for using a substitution mobile router or communication device in a specific place, thus allowing a better network service. For this purpose, experimental environment and platform have been set at LAAS for running controlled experiments on wireless networks, and an analysis methodology has been designed. The platform has been set up in an anechoic room to avoid external perturbations, and in order to control the signals generated. Large experiment campaigns have been run. The analysis of the traces captured is made using an SVM (Support Vector Machine) based method. Results exhibit that by looking at very few signal parameters is enough to predict the evolution of the physical air environment, and of the service provided by the network at the network digital level. It also exhibits that this is possible to predict on a 500 ms basis the evolution of the network quality of service.

This work is described in the paper "Predictive Estimation of Wireless Link Performance from Medium Physical Parameters Using Support Vector Regression and k-Nearest Neighbors" that has been accepted for presentation at the TMA (Traffic Monitoring and Analysis workshop) to be held in London on April 14th, 2014.

Networks considered in the RESCUE project are wireless (at least for the substitution routers) and dynamic (potentially highly dynamic). Indeed, this last decade has seen an increasing interest for wireless communications. With the current use of smart-phones and tablets coupled to the rise of the Internet of Things, the number of mobile terminal nodes in networks will significantly change the way we manage them. Indeed, these wireless networks are more and more dynamic, especially concerning topology. Therefore distributed network control, and traffic management are of increasing complexities: How to face a topology fragmentation in the network? Which considerations to take when lightening a crucial node? Networks will need to be autonomous regarding the service they are providing to users and their resilience. Mobile substitution networks will then be of essential importance in a very short future. Thus, networks need a way to self-evaluate their wireless performance and deploy routers at the best appropriate locations. We then introduce the concept of System Development Index (SDI), which allows a system to judge how well it performs. The SDI is a generic index. In the framework of the RESCUE project, we specifically integrate signal level parameters for defining the SDI index. We then provide methods and algorithms, which are based on events collection and distributed mining to analyze the evolution of this index.

In our instantiation, nodes observe events, summarize these observations and can share them with its neighbors to infer properties on the SDI. We illustrated it by simulation, considering different scenarii under NS3. We evaluated the evolution of the SDI and analyzed its possible underlying reasons for one scenario and analyze error estimation for various network properties.

Based on this SDI, it is then possible to infer when substitution routers are needed, and at what place.

This work is described in the paper "Event Based Performance Evaluation and Analysis of Dynamic Wireless Networks" that has been presented at the CNSM conference (conference on Network and System Management) held in Zurich on October 14th-18th, 2013. In this report an extended version of the paper is included.

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.**

I. 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.

II. Machine learning algorithms

A. 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.

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

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

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.

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

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

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 $\alpha_i^* \in [0, C]$.

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

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 $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$. The Gaussian kernel is parametrized by γ ($\gamma > 0$) which impacts the generalization capability of the regressor among other things.

B. 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 $(\sum_{i=1}^{n} |x_i - y_i|)^{\frac{1}{p}}$.

III. Experimental platform and dataset

A. 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.

B. 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

TABLE I: 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};...
$Dataset1$ (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}
$Dataset2$ (2661 vectors)	{10;-20};{10;-17};{10;-15};{10;-13};{20;-20};{20;-17};{20;-15};{20;-13}
$Dataset3$ (1330 vectors)	{10;-20};{10;-17};{10;-15};{10;-7};{10;-5};{20;-20};

protected context minimizes the uncontrolled parameters of our communication.

C. 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 chip-set 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.

1) 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).

2) 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.

D. Experimental protocol

1) 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 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.

2) 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 I 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 I. These datasets differ by the quantities of samples they are made of, and consequently by the level of knowledge they represent.

E. SVR features definitions

1) 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: $P_i = \{p_1, ..., p_n\}$.

2) 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 $R_i = \{RSS(p_1), ..., RSS(p_n))\}$.

3) 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 $p_k \in P_i$.

4) 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.

5) 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 $p_k \in P_i$.

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

A. 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.

1) SVR: SVR algorithm has been used with RBF as a kernel function. As section II-A 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.

2) 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].

3) 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.

B. Estimation performance

To evaluate the estimations, two methods are used.

1) 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 $MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y_i} - Y_i)^2$.

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.

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

(a) Scores and pertinence of the estimations.

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

(b) Average delays observed for the training and 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 training (s)						Time used for estimation (s)					
		SVR			k-NN			SVR			k-NN		
		$D1$	$D2$	$D3$	$D1$	$D2$	$D3$	$D1$	$D2$	$D3$	$D1$	$D2$	$D3$
1	ATH_RSS	5.39 *(40)*	1.40 *(2)*	0.36 *(0.4)*	0.048 *(2)*	**0.023** *(0.1)*	**0.012** *(0.1)*	1.52 *(10)*	0.78 *(6)*	0.40 *(3)*	1.13 *(10)*	0.63 *(1)*	0.39 *(0.6)*
2	$LECR\_RSS$	41.27 *(300)*	11.71 *(7)*	3.12 *(2)*	0.048 *(1)*	**0.023** *(0.2)*	**0.012** *(0.1)*	1.58 *(20)*	0.81 *(10)*	0.42 *(3)*	0.61 *(3)*	0.44 *(0.8)*	0.29 *(0.6)*
3	$LECR\_NOISE$	5.17 *(10)*	1.38 *(1)*	0.36 *(0.8)*	0.051 *(6)*	**0.023** *(0.2)*	**0.012** *(0.1)*	1.47 *(20)*	0.76 *(10)*	0.42 *(4)*	0.96 *(30)*	0.35 *(0.9)*	0.08 *(0.3)*
4	$LECR\_SNR$	11.54 *(6)*	3.87 *(4)*	1.35 *(2)*	0.048 *(4)*	**0.023** *(0.2)*	**0.012** *(0.1)*	1.38 *(30)*	0.70 *(10)*	0.36 *(3)*	0.74 *(60)*	0.45 *(0.8)*	0.19 *(0.3)*
5	$ATH\_RSS + LECR\_NOISE$	**4.50** *(4)*	**1.15** *(2)*	0.30 *(0.2)*	0.048 *(0.6)*	**0.023** *(0.1)*	**0.012** *(0.1)*	**1.34** *(9)*	**0.67** *(10)*	**0.35** *(8)*	0.32 *(10)*	**0.15** *(0.3)*	**0.06** *(0.1)*
6	$LECR\_RSS + LECR\_NOISE$	4.72 *(9)*	1.23 *(2)*	**0.31** *(3)*	**0.046** *(0.5)*	**0.023** *(0.1)*	**0.012** *(0.08)*	1.38 *(1)*	0.70 *(4)*	0.36 *(3)*	**0.27** *(0.4)*	**0.15** *(0.2)*	0.08 *(0.1)*

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

C. Estimation results

Table II 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 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.

D. Training and estimation time performance

Table II 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 factor of 2 when changing from $Dataset1$ to $Dataset2$ and then from $Dataset2$ to $Dataset3$. According to section II-A, SVR forces the estimated function to be within

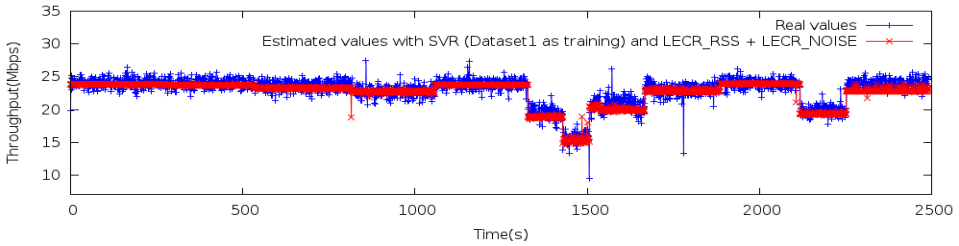


Fig. 1: Throughput estimation results obtained with the $LECR\_RSS+LECR\_NOISE$ metric compared to the real throughput.

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.

V. 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.

VI. 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.

Event Based Performance Evaluation and Analysis of Dynamic Wireless Networks

Denis Carvin[1,2a], Guillaume Kremer[1,2b], Philippe Owezarski[1,2c], Pascal Berthou[1,2b]
[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, [a]INSA, [b]UPS, [c]LAAS, F-31400 Toulouse, France
Email: {carvin, kremer, owe, berthou}@laas.fr

***Abstract*—This last decade has seen an increasing interest for wireless communications. With the current use of smart-phones and tablets coupled to the rise of the Internet of Things, the number of mobile terminal nodes in networks will significantly change the way we manage them. Indeed, these wireless networks are more and more dynamic, especially concerning topology. Therefore distributed network control, and traffic management are of increasing complexities: How to face a topology fragmentation in the network? Which considerations to take when lightening a crucial node? Networks will need to be autonomous regarding the service they are providing to users and their resilience. Mobile substitution networks will then be of essential importance in a very short future. Thus, networks need a way to self-evaluate their wireless performance and deploy routers at the best appropriate locations. In this paper, describing our work in the framework of the RESCUE project, we introduce the concept of System Development Index (SDI), which allows a system to judge how well it performs. We then provide methods and algorithms, which are based on events collection and distributed mining to analyze the evolution of this index. In our instantiation, nodes observe events, summarize these observations and can share them with its neighbors to infer properties on the SDI. We illustrated it by simulation, considering different scenarii under NS3. We evaluated the evolution of the SDI and analyzed its possible underlying reasons for one scenario and analyze error estimation for various network properties.**

I. Introduction

Nowadays, one can say that wireless mobile networks have definitely invaded our daily lives. When looking at forecasts from Cisco [1], mobile traffic should increase 7.5-times in the 4 next years. While the service quality will be of primer interest (two-thirds of this traffic will be dedicated to video), provider systems will be tricky to manage. Managing wired networks to sustain unpredictable traffic is still a complex and unsolved technical domain. The problem raised by wireless networks will be even more complex: in particular, wireless infrastructures will experience dynamic topologies depending on terminals position and needs, thus adding a level of magnitude of complexity. Indeed, complex phenomena will arise with the Internet of Things where random interactions might generate new kinds of traffic. As a result, system managers will not be able to quickly analyze the underlying reasons that affect their networks' behaviours, and so, to launch appropriate corrective measures. Thus, this task needs to be mainly delegated to systems themselves, for providing fast autonomous adaptative actions when required (when detecting events leading to network QoS decreases, or even better when predicting such QoS decreases). In this paper we provide a reference model for an observer to assess a system and then conduct an assessment-centric analysis. While this model could be applied to systems in general, we consider self-assessment and self-analysis capabilities for dynamic wireless network of collaborative nodes. More specifically, our work is focused on the service provided by the network layer. In this particular case we will have to face two majors difficulties: (1) nodes only have partial information on the network state, (2) the level of service relies on a wireless medium which is complex and not reliable by nature. Concerning self-assessment, partial information forces nodes to agree on a value based on their local information. Also, this value is time varying and the problem is from the class of distributed consensus. Concerning self-analysis, nodes will have to search and share the information they dispose to understand the law followed by the consensus value. Then, they will have to determine whether they are responsible for its evolution. This could be seen as a distributed data-mining in uncertain context problem, uncertainty being harshened by the medium kind. Therefore, the remainder of this paper is structured as follows. The following section describes the related work on data-mining applied to network management and distributed consensus. We introduce in section III our general assessment and analysis framework for multi-agent systems. In section IV, we instantiate this framework in the case of wireless dynamic systems and lead an off-line analysis through an example scenario. In the fifth section a distributed assessment algorithm is provided for our network, we also evaluate it under various network conditions. We will conclude on future work in a last section.

II. Related Work

A. Knowledge Extraction and Network Analysis

Understanding and Managing wireless network is one of the operator concern. Orange Labs have shown interest on the optimal deployment of wireless substitution network [2]. On its side, AT&T Labs worked on data-mining to analyze its own wireless infrastructure [3]. Data-mining applied to networking has already been investigated, in particular by the security community which is quite fond of this angle. The approach is applied to network intrusion detection but also in traffic monitoring and anomaly detection. The main idea is to lighten network manager task by removing false positive alert. In [4] the authors mentioned that data-mining was a valuable tool

but which was not about making human analysis unnecessary, specifically in the attribute choices. Then data-mining are useful to construct new rules. For example, technical results can be found in [5] where Casas & al. demonstrated the efficiency of clustering techniques to detect traffic anomaly and construct new filtering rules without knowledge. Also, understanding cause and effect between network events is not the stronghold of security. We found in [6]–[8] analysis concerned by the understanding of network behavior. In [6] authors highlighted the sources of TCP reset anomalies. The field of wireless communication is investigated in [7] where the key characteristics of the traffic are captured on several base stations to optimize their coordination. Authors showed a significant enhancement on the downlink delay performance by clustering users in profiles. Finally authors of [8] focused on the relation that can exist between user experience and the network quality of service. The studies was lead on a set of mobile users and explained the relation between server response time, round time trip and user satisfaction. While [5], [7] have brought methods to extract information, [6], [8] have tailored their studies toward a very specific goals. These two approaches need to be linked by a common objective which is the network performance. Also, narrowing extracted information down to assessment will allow one to take up the issue of partial information. Indeed, despite the efficient work cited above, it only considers an omniscient and centralized approach because of their domain complexity. Therefore we insert our work in between, with the motivation to only extract clues on a system relatively to its assessment. The assessment needs to be shared among all the nodes, which leads us to the distributed consensus problem. Since our assessment method relies on an average over nodes, we will focus on the average consensus problem.

B. Distributed Consensus And Average Sharing

In distributed algorithms, when agents need to agree on a value, it remains on the well known consensus problem. This class of problem has been deeply studied from various angles (termination, fault tolerance, etc...). A subclass of this problem is called average consensus, where each agent i keeps a value v_i. The consensus for agents is to find the average x of the kept values. Many algorithms have been designed, this is perfectly illustrated in [9]. Every algorithm is a trade-off between time convergence, memory used by each agent and the number of exchanged messages. Among the ones that require few communications and computational resources, the most famous are : the maximum degree weight given in [10], the metropolis weight given in [11]. In these schemes, each node proceeds iteratively, computes values and sends them to its neighborhood at each step. These linear algorithms both consider that the dynamic of the average is greater than the convergence time for a connected network. The estimation $\widehat{x}_i(t)$ of x for node i at step $t+1$ is given by :

$$\widehat{x}_i(t+1) = \alpha_i.\widehat{x}_i(t) + \sum_{j \in N_i(t)} \alpha_j.\widehat{x}_j(t) \quad and \quad \widehat{x}_i(0) = v_i$$

where $N_i(t)$ is the neighborhood of i at step t and α_k depends on the algorithm:

Maximum degree weight algorithm for N agent:

$$\alpha_k = \frac{1}{n} \quad if \quad k = i \quad and \quad \alpha_k = 1 - \frac{|N_i(t)|}{n} \quad otherwise$$

Metropolis weight algorithm:

$$\alpha_k = \frac{1}{1 + max(|N_i(t)|, |N_j(t)|)} \quad if \quad k \neq i$$

$$and \quad \alpha_k = 1 - \sum_{k \neq j} \alpha_k \quad otherwise$$

Keeping in mind the introduced related work, we can now detail our assessment and analysis framework of systems.

III. An Assessment and Analysis Framework for Multi-Agent Systems

A. System Development Index and Assessment

1) A Multi-Agent Systems Model: Inspired from the Multi-Agent Systems theory, our framework considers a System as a set of agents. An agent is an entity which owns interfaces to interact with its environment and specifically with others agents. An agent also owns interfaces to observe events or interactions that occurred to him or to its neighborhood. Each agent has an utility value that represents its wellness over time. We defined utility values in the real interval $[0, 1]$, where 0 is a worst case and 1 is a best case. The utility value of an agent can also be seen as its percentage of satisfaction driven by a possibly unknown utility function. Each agent is able to store a set of observed interactions as well as its satisfaction history. The formalism we used to describe these observations is detailed in the next section.

2) System Ideality and System Development Index: When considering system assessment, our main axioms are the following:

- Systems where all agents are 100% satisfied are ideal.
- Systems where all agents are 0% satisfied are not ideal.

Therefore we can summarize the level of ideality of a N-agents system in a N-dimensional vector of utility values.

- Let S be an N-agents system
- Let $u(t)$ be the utility vector of agents in S at time t
- Let $u_i(t)$ be the utility of agent i in S at time t
- Let z $\in [0,1]^N$ such as $z_i = 0 \quad \forall\ i \in [1, N]$
- Let o $\in [0,1]^N$ such as $o_i = 1 \quad \forall\ i \in [1, N]$

A SDI for a N-agents System S is a function from $[0,1]^N$ to $[0,1]$ that satisfies both of our axioms, thus the space of SDI function for S is given by:

$$\{g : [0,1]^N \to [0,1] \mid \quad g(o) = 1\ and\ g(z) \neq 1\}$$

The system S is said ideal for a given SDI g at time t if and only if :

$$g(u(t)) = 1$$

Thus in our framework, system ideality is related to a point of view or function. The classical form of SDI is given by

$$f(s) = \sum_{i=1}^{N} \gamma_i.u_i \quad with \quad \sum_{i=1}^{N} \gamma_i = 1$$

The value of γ_i could be based on the pricing policy of a system manager who gives preferences to some users classes. For a selfish agent i, the system will be ideal as far as its own satisfaction equals 1. In the latter case $\gamma_j = 0$ for $j \neq i$ and $\gamma_i = 1$. One can also base a SDI on a distance between the satisfaction distribution and the perfect distribution where all agents are fully satisfied. The main point is that a SDI is an index that one wants to follow. For an external observer, this index can just be studied. In the case of a concerned observer like a system manager, this index will be tracked in order to be maximized. Finally if the observer is an agent, the index indicates how well the system in which it evolves is ideal given its point of view.

3) SDI Estimation and Analysis in Various Systems: Computing the value of the SDI in real time is not trivial for systems with a large number of nodes. Indeed one needs to have access in real time to the utility value of each agent to be able to compute its exact SDI. From an agent point of view, it means that he should collaborate with others, which is not always the case. For the same reasons, accessing the necessary information to understand the evolution of the SDI is not always possible. We can thus study the SDI with different angles which depend on omniscience, and interactivity (off-line vs on-line):

Omniscient Off-line Studies: This kind of study considers the whole set of information contained by the agents and analyzes it passively. The aim here is to identify behaviors, cause-and-effect regarding the evolution of the SDI for a whole system. It allows one to retrieve knowledge on the underlying reasons that drive the assessment of a system under a given point of view (SDI). Consequently, these studies are preliminary studies.

Partially informed Off-line Studies: In this case the information considered is only contained by a subset of agents (or a unique agent) under a passive analysis. The main goal of these studies is to characterize the possible conclusion that a subset of agents could draw with partial information.

On-line Studies: While for Off-line study it is possible to be omniscient, in the case of On-line studies, the information will always be considered as partial, since the agents can not know the whole state of the system. Thus they can not be sure of the consequences of their actions. That is the reason why this type of analysis better requires preliminary off-line survey and calibration.

B. Event Based SDI Analysis

In order to analyze the evolution of a system, we consider its initial state and its succession of events. We have previously described our model such as a set of agents having interfaces through which they observe events. In this section we will detail the formalization of these observations and the way to analyze them.

1) Observation and Event Definitions: An observation can be seen as tri-dimensional point. More specifically an observation has a time dimension, an observer (or agent) dimension and an event dimension where events are also multidimensional. An example of table is illustrated in table III-B1.

Time: The time dimension is crucial since we want to study and manage the temporal evolution of the SDI. Time is considered continuous, second(s) is the principal unit. We keep only one time representation which is called *#Time*. In our implementation, we considered time as a float value with the experiment start as the origin.

Agent: Agents are the main entities of our systems they interact each others, observe, analyze and make decision. They will have a unique identifier *#Agent*.

Event: An event is a *perceptible* modification of the system state. Combining the initial state and events, one can trace a partial history of a system. As specified above, an event is a multidimensional object identified by a primary key *#Event*. It can be represented by a frame where the first field is the event type (*eType*) which determines the validity and the meaning of the following ones. An Event can occurs several times and be observed at different moment by distinct agents.

(a) Example of Observations Table

Observations		
#Time	**#Agent**	**#Event**
float	*int*	*int*
1.2	0	0
1.25	1	1
1.255	0	1
1.3	1	0

(b) Example of Event Table

Events					
#Event	**eType**	**eSource**	**eSpeed**	**eLength**	...
int	*string*	*int*	*float*	*int*	...
0	'Move'	0	3.0	-	...
1	'Packet'	1	-	1500	...

TABLE I
*

In this scenario, we have two agents, each of them produces an event. Each event is observed by both agents. Agent 0 moves at time 1.2s while agent 1 sends a packet at time 1.25s. Agent 0 observes the same packet 5ms later while agent 1 realizes that agent 0 has moved at 1.3s. The two events have a field in common which is the agent id that produced the event (*eSource*). The two events have also distinct fields: an event of type *Packet* has a length (*eLength*) whereas a *Move* has a speed (*eSpeed*)

2) Constructing Observation Features: The idea behind an event based SDI analysis is to link the evolution of an SDI to the evolution of observations features. We call feature a property of an observation cluster. Thus, we will create clusters of observations, compute some cluster properties that vary over time and then study the association between these properties and the SDI. As a result, when proceeding to an SDI analysis,

one wants to define three important things : (1) An algorithm to define clusters, (2) distance functions between observations and (3) the properties to observe. Therefore we give in table III-B2 examples of canonical distances that one could use to build a distance between observations.

Dimension	Distance(o1,o2)
Time	abs(o1.#time-o2.#time) abs(HoD(o1.#time)-HoD(o2.#time)) abs(DoW(o1.#time)-DoW(o2.#time))
Agent	2-norm(o1.#agent.position,o2.#agent.position) o1.#agent.nbHop(o2)+o2.#agent.nbHop(o1)) RTT(o1.#agent,o2.#agent)
Event	LevenshteinDist(o1.#Event.eType,o2.#Event.eType) card({field | o1.#Event.field≠'-' ⊕ o2.#Event.field≠'-'})
Generalized	Same(f(o1),f(o2)) where Same(x,y)=0 if x=y ; ∞ otherwise example : f(x)=(x.#agent,seconds(x.#time))

TABLE II
*

This table gives canonical distances for each dimension to be used when grouping observations. For time dimension we suggest the use of Hour of Day (HoD) or Day of Week(DoW). For event dimension, the string comparison of type name is possible (Levenshtein or edit distance). In this paper, we will use a generalized distance (Same) based on lambda functions (in our case key comparison)

Regarding time, one can express naturally the distance between two timestamps as a simple difference of the values in seconds. Nevertheless, depending on the studied system, it could be meaningful to use seasonal distance like the difference between hour of day or day of week. When considering agents, the natural way to evaluate distance is to use geographical positions. This last approach might not make sense if agents are software entities in a same physical system. Then, analysts might want to define other distances like the proximity of their state or the number of hops in the case of networks. Distance between events are less obvious to determine. However it is still possible to create generic metrics based on the string distance between their type names, their number of common fields or the value of their fields.

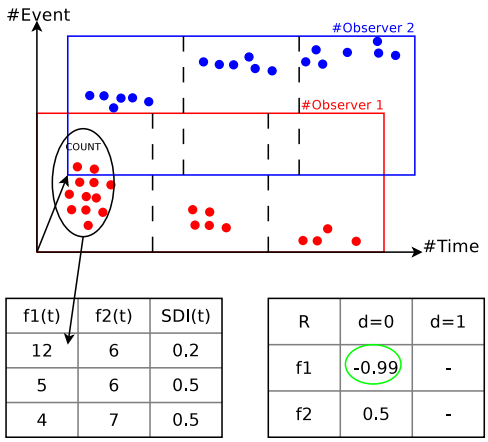


Fig. 1. Features and SDI Analysis

3) Temporal Correlation Between Feature and SDI: In this paper, we will only construct features in a supervised way using aggregation over observations. After having grouped observations (for example by observers and/or type of event) we construct subgroups by time intervals. We then apply an aggregate function (such as count, or average over a field) to build time series of features. For illustration purpose, a trivial but significant example of feature is the number of events observed by an agent during a unit of time like illustrated in the figure 1. Following this process we can construct a set of time series of features $\{f_1(t), f_2(t), f_i(t), ..., f_p(t)\}$. Once these time series are built, we can study their delayed correlations with a SDI $g(t)$ over a period of time. We can thus determine the features that might have driven the SDI evolution during this period.

- Let $t \in [1, T]$ be a period of time
- Let $\{f_i(t) \mid i \in [1, P]\}$ be the associated time series
- Let $g(t)$ be an SDI
- Let $d \in [1, D]$ be a delay

We defined the matrix R: $P \times D$ of $r_{i,d}$ as the delay correlation matrix where $r_{i,d}$ is the correlation coefficient between $f_i(t)$ and $g(t+d)$. The final goal is to find the coefficient in the matrix that have the highest magnitude in order to highlight plausible causes of the SDI evolution.

IV. Application to Dynamic Wireless Networks

So far, we have presented a framework to assess systems and to analyze the underlaying factors of this assessment. In this section, we will use this framework to assess dynamic wireless networks. This instantiation narrows the general case to systems where agents are not malicious, use the same SDI functions and share a protocol to exchange information on this SDI.

A. Considered Network Scenarii

The considered system is a wireless mobile ad-hoc network. We implemented it under the Ns3 simulator . Each node has its own mobility model and dynamism. Nodes have a unique wireless interface, might run an UDP server, and instantiate severals UDP On/Off Constant Bit Rate traffic sources. Each UDP source has a destination among the set of server nodes. A source has a fixed data rate and packet size. Duration of activity phasis follows a uniform distribution with fixed bound. We used Ns3 YansWifi Model. Controllers are set in ad-hoc mode and use the adaptive auto rate fallback algorithm without any quality of service. Routes are discovered through the use of AODV. The full list of configurable parameters is described in table IV-A, while a scenario illustration is given figure 2.

B. Framework Instantiation

In this network, nodes are the agents. Each node has a satisfaction function based on the delay it experiences during its communications. It interacts with its environment essentially by its moves and its communications. It can observe others communications and record its own events.

Network Parameters	
N	Number of Nodes
randSeed	Pseudo-random generator initializer
For each node	
X	Initial position on X axis
Y	Initial position on Y axis
hasServer	Implement an UDP server
nbSrc	Number of UDP source
dataRate	Source data rate
pktLen	Packet size
onTime	Min-Max On period duration of sources
offTime	Min-Max Off period duration of sources
noiseFig	Noise figure of the Wifi receiver
mobiModel	Mobility Model (Constant, Random Waypoint, Random Walk)
speed	Min-Max Speed
pause	Min-Max duration of a stable position
xRange	Min-Max position on X-axis
yRange	Min-Max position on Y-axis

TABLE III
*

We can configure several parameters in our Ns3 environment. In the network, each node has an initial position and can move in a defined area with a tuned mobility. We can also influence the traffic matrix by configuring UDP sources and servers

***Event, Agent and Observations**:* Since Ns3 is an event based simulator, it offers interesting properties to instantiate our framework. Among them, its tracing system allows the easy implementation of event observations. In our instantiation, time is a float where the origin is the beginning of the simulation, each observer is an agent, whose id is derived from its IP or MAC address. We have defined various types of events, but we can sort them into two main classes: (1) Packet events observable from different nodes (2) Others events, internal to an agent and only accessible by this agent. The latter are : nodes moves, routing table attributes modifications, errors and drops. Table IV delivers further details on the different instantiated types of events.

TABLE IV
EVENT TYPE DESCRIPTION

Type	Information
Packet	Packet capture in promiscuous mode with radiotap header
Rtam	A routing table attribute is modified (number of valid entries, longuest path...)
Move	Speed modification along at least one axis
Ipv4Drop	Packet Drop for a routing reason
PhyRxError	Frame has been received unsuccessfully
PhyRxDrop	Frame dropped during reception
MacTxDrop	Packet dropped before being queued for transmission
MacRxDrop	Packet dropped after the Physical layer
MacTxDataFailed	Data packet transmission failed at mac layer
MacTxRtsFailed	RTS transmission failed at mac layer
MacTxFinal DataFailed	The number of consecutive MacTxDataFailed has reach a threshold
MacTxFinal RtsFailed	The number of consecutive MacTxRtsFailed has reach a threshold

***Satisfaction and SDI**:* In our particular case, the SDI is computed from the observations themselves. We have chosen a fixed aggregation time of 1 second to analyze the network events and construct features. This value is small enough to follow the SDI evolution while sufficiently large to smooth small wireless dynamics. The network assessment is given by the following formulas. Each packet that an UDP source has generated is scored. The scoring function is :

$$score(d) = max\Big(0, \frac{threshold - d}{threshold}\Big) \quad d = delay(p)$$

The score linearly decreases when the delay increase between 0 and a given threshold. It equals 1 for a null delay and 0 if the delay is greater than a threshold (or if the packet is lost). We set the threshold to the arbitrary value of 10ms. A delay associated to a Packet is the timestamps difference between its first observation on the wireless medium and its first observation by its destination. This score is a QoS metric that could be link to the user satisfaction like [8] did. The satisfaction of node i for the interval T is given by the average score for packets that have been generated by i during the interval T:

$$Sat_i(T) = |D|^{-1}.\sum_{d \in D} score(d)$$

$$Sat_i(T) = 1 \quad for \quad |D| = 0$$

$$D = \{delay(p) \mid p.ipSrc = ip(i) \ \cap \ p.time \in T\}$$

The SDI we choose is a simple average of satisfaction over nodes. Thus the SDI for a network of N nodes associated to the interval T is given by:

$$SDI(T) = |N|^{-1}.\sum_{i=1}^{N} Sat_i(T)$$

C. Analysis of Dynamic Wireless Networks

For understanding purpose, we will illustrate the analysis of the scenario given in figure 2. This analysis will be off-line and the observation will be omniscient. As we specified above, we have constructed our features based on aggregation function. Mainly, we grouped observations by observer and by second. We have designed more than twenty features by nodes, table V details the most relevant ones for this scenario.

Name	Information
AvgnbGateway	Average # gateway in the routing table
AvgnbValid	Average # valid entry in the routing table
CountPhyRx	# received frame
CountAllRetry	# frame having a retry flag
CountMyRetry	# transmitted frame with a retry flag
CountMyIpFlow	# local distinct IP destination
CountMyUdpSrc	# local active UDP sources
CountAllFlow	# IP flow going through the local node
CountPhyRxError	# PhyRxError events
CountPhyRxDrop	# PhyRxError events
CountDropRouteErr	# IPv4Drop events for a route error reason

TABLE V
FEATURES DESCRIPTION

Most relevant constructed features related to the scenario. Each features is related to an observer (called local node). The # stands for "*number of*"

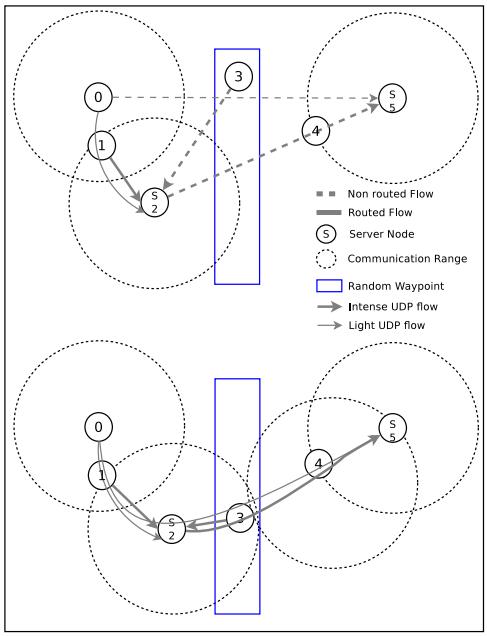
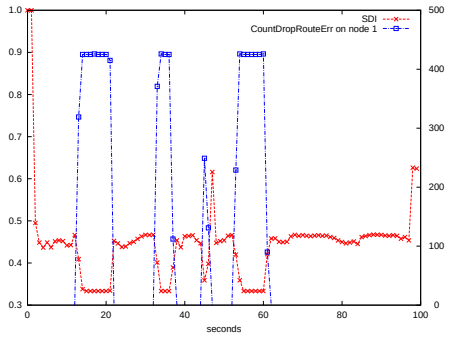
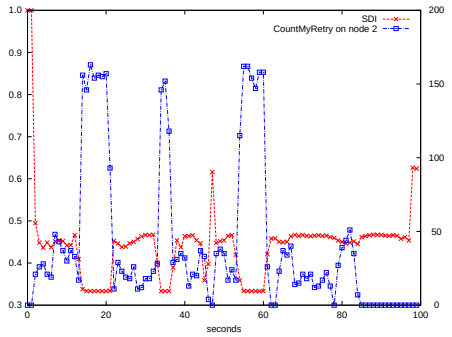


Fig. 2. *
In this scenario, Node 2 and 5 are UDP sinks. Node 3 is mobile. On the top node 0 and 1 might overload node 2 while at the bottom, route toward 5 is down.

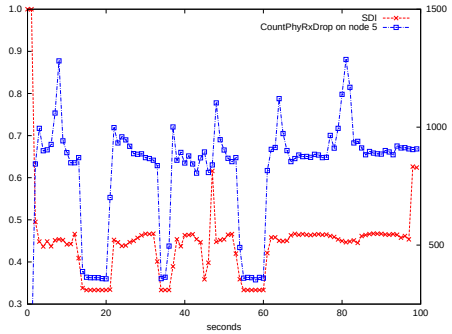
At the beginning of the scenario, all the sources where off, thus all nodes were fully satisfied. Traffic sources started to transmit from second 2 when the SDI brutally decreased. Then, for every significant move of node 3, IP routes are lost or recovered, impacting significantly the SDI. When routes are up, fluctuations can be explained by the delay variations. When nodes 2 experiences some difficulty to transmit, its number of retry will increase and impact the delay. Even if only few packets are concerned, this might have a significant impact on the source satisfaction if these packets are the only ones sent by the source node. Since our SDI takes every nodes in consideration, without any regards on their source volume, we can see an effect on the SDI. After having computed the delayed correlation matrix we found high values for the three features illustrated in figure 3. *CountDropRouteErr* on node 1, *CountPhyRxDrop* on node 5 and *CountMyRetry* on node 2 scores are respectively -0.92, 0.79, -0.88. In figure 3(a), we clearly show that the main fluctuation of the sdi is due to a routing error. Indeed, the node 1 can not find a route to node since node 3 has left the path. The retries experienced by node 2 is detailed in figure 3(b) it impact the SDI when the route is up with a bad communication link between 2 and 3. At first, one can think that transmission retries of node 2 are introduced by the physical drops on node 5. In fact, those events are negatively correlated. Indeed, these nodes can not reach each others due to their relative distances, since the number of drop is much greater than the retry, it might come from the fact that node 5 could still be in the carrier range of node 2. Figure 3(c) confirms that node 5 does not drop packet for low SDI, since node 2 does not sent them because of routing errors.



(a) CountDropRouteErr on node 1



(b) CountMyRetry on node 2



(c) CountPhyRxDrop on node 5

Fig. 3. Temporal Evolution of SDI Regarding 3 Features

In this case, we led an off-line analysis based on the real value of the SDI. Using simple features based on event counts, we were able to diagnosis the sources of the SDI fluctuation. By construction, these features can be computed in real time by nodes and exchanged to analyze the situation.

V. A Collaborative SDI Estimation

Agents consider their own observations to estimate the SDI value and communicate with their neighbors. We suggest an estimation based on existing consensus algorithm. At this point, network layer needs to estimate the satisfaction of upper layers. In our case this comes to evaluate the end-to-end delay experienced by the local application. Thus, in order to estimate the SDI, we first estimate the delay, this estimation will serve to approximate the local satisfaction. Finally we exchange this local satisfaction to have an estimated SDI.

A. Delay Estimation

The end-to-end delay for a packet has already been defined above as the temporal difference between source and destination observations. Each time a packet needs to be forwarded by a node, it waits for a medium access. This time is difficult to predict in our case. Indeed, medium access will be impacted by the level of noise, the number of neighbors, their proximity, their load and the level of interference they produce. As a result, we will approximate the average delay by a sum of average medium access times. Our approximation will take into account the traffic matrix so that for an interval T, the estimated average delay that a packet can experience when leaving a node i is:

$$\widehat{D}_i(T) = L_i(T) + \sum_{j \in G_i(T)} \rho_j(T).\widehat{D}_j(T-1)$$

$$with \quad \widehat{D}_i(0) = L_i(0)$$

$L_i(T)$ is the average local medium access time, $G_i(T)$ is the set of gateway used by node i during T, $\rho_j(T)$ is the percentage of data traffic sent/forwarded by node i during T that should be forwarded by node j. For each packet, the local processing time equals 0 if the local node is the destination, it equals the threshold value if the packet is dropped, in other cases it is the time between the first observation of the packet from the upcoming link, and the last observation of the packet on the outgoing link. Given a packet arriving at node i, its expected delay is at least the local link process. Depending on its destination, this process time will be added to the expected delay of its destination. The expectation is materialized by ρ, which is a percentage of traffic. The computed average delay for a node will depend on the previous computed value of its neighbors. This implies that nodes have to regularly communicate to update the value of their delay.

B. Local Satisfaction Estimation

Once a node is aware of the average delay it experiences, it is capable to estimate its satisfaction. We approximate the satisfaction by assuming that expectation and scoring function can commute. That is to say (with E_D the expectation function over a set D of delay):

$$E_D(score(d)) \approx score(E_D(d))$$

The approximation error is null when all the observed delays are under the threshold, since the scoring function is linear on the interval [0,threshold]. Thus, we can find a threshold where the approximation can be acceptable. Therefore, we use the average delay $\widehat{D}_i$ to approximate $Sat_i(T)$

$$\widehat{Sat_i}(T) = score(\widehat{D}_i(T)) \approx Sat_i(T)$$

C. SDI Estimation

Since we have defined our SDI as an average, the SDI estimation problem is in fact a dynamic average consensus problem. In our case, the average evolves over time as well as the topology, which is not a fortunate case for previous algorithm. However, despite our problem complexity, satisfaction of nodes over a network are linked in some ways and their dynamics rely on events. Thus the values that composed our average are related and their temporal evolution are driven by the network itself. Therefore we derive an algorithm from existing ones to estimate the SDI value, then we study the impact of satisfaction fluctuation and dynamic topology. In our case, we want to estimate the SDI, which is a time-varying average of the satisfactions. We modified the scheme presented in II-B to suggest the following iteration:

$$\widehat{SDI}_i(t+1) = \alpha_i.\widehat{Sat}_i(t) + \sum_{j \in N_i(t)} \alpha_j.\widehat{SDI}_j(t)$$

$$\widehat{SDI}_i(0) = \widehat{Sat}_i(0)$$

Also if the local satisfaction is known, the real value can be used instead. In this scheme, the satisfaction term allows the consideration of the local satisfaction while the SDI term permits the estimation propagation over the network. The main difference is introduce by $\widehat{Sat}_i(t)$. This term introduce the variability of the satisfaction over time, which was not the case in previous algorithms. The value of α_k could be choose from the metropolis weight or the maximum degree weight.

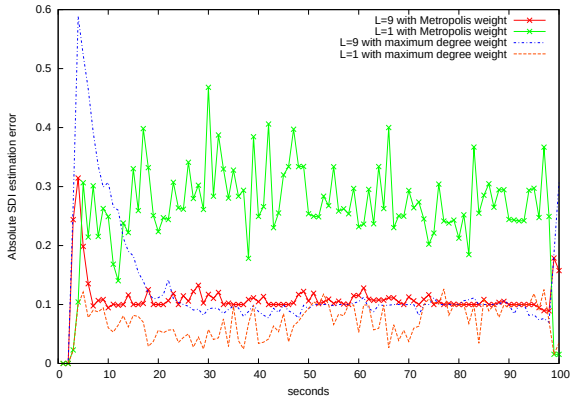
D. Local Estimation Results

In order to assess the accuracy of our local SDI estimation, we compare estimated SDI values with the real SDI. We conducted 432 Ns3 simulations run to measure the impact of networks characteristics like radius, load and dynamic. We fixed the number of node to ten. Each scenario combines different values of the following parameters : Number of source, Initial average distance, mobility and random seed like illustrated in table V-D

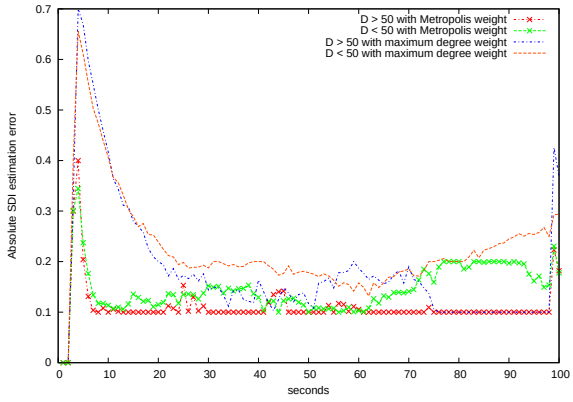
Network properties					
Number of Nodes	10				
Source Data Rate	1 Mb/s				
Packet size	1470 Byte				
Source duty cycle	1 (always On)				
Min-Max Speed	5-7				
Random seed	0,1,2				
Number of source (L)	1,4,7,9				
Number of server	10-L				
Initial spacing (d)	20,45,65,75				
Mobility Model	Constant	Random Walk		Random Waypoint	
Area size	-	dxd	d/2xd/2	dxd	d/2xd/2
Pause duration	-	-	-	10, 25, 65	10,25,65

Based on the data set obtained from these simulations, we were able to study the behavior of both weighting algorithm for the average consensus : Metropolis algorithm and Maximum weight algorithm. We compare their properties in different situations. In the first situation we considered a constant topology and two levels of load which were L=1 and L=9. In the second case we considered an heavy load (L=9) with two levels of mobility. For the first level of mobility, nodes were able to move in a constraint area (D $\leq$ 50), for the second level nodes were moving in a wider area (D>50). For all scenarii, we computed the absolute value of the difference

between the real SDI and the estimated SDI for each node and iteration. To study the convergence rate and the evolution over time, we computed the average of this error. In figure 4 we plotted the obtained average for all the scenario in the considered cases.



(a) Impact of load under a constant topology



(b) Impact of mobility under an intense load

Fig. 4. Average error for estimated SDI under several network profiles. For clarity purpose, we did not plot the interquartile distance. For information, their values were always under 0.20 for all the curves.

As expected, the initial error tends to be important since nodes have no idea of their neighbor satisfactions. Considering figure 4, because the SDI does not have a constant value, the convergence is not reached. However, the algorithm tends to reduce the estimation error over time. This error converges toward 0.1. In 4(a) like in the case of a constant value, metropolis weight seems more reactive than the maximum degree weight, which, in the case of low traffic might be seen as an over-reaction. When regarding figure 4(b), under an heavy traffic, algorithm appears to better perform under a very dynamic topology. The reasons come from the SDI itself. When the network is too dynamic, routing protocols do not perform well. As a result, the local satisfaction are sufficient not to make mistake in the SDI estimation.

VI. Conclusion and Futures Works

The invasion of mobile communication in our network have definitely changed their level of complexities. The terminal heterogeneity have multiplied traffic profiles, users are dynamic as well as the topologies. Thus, the management task needs to be mainly delegated to network. In doing so, networks need to evaluate themselves and understand the way they behave. In this paper, our contribution was two-folds. First we introduced a reference model for an observer to assess multi-agent systems and conduct an analysis focused on this assessment. We consider that evaluation are temporal scores, besides we collect event observations to construct time series of features. Our analysis is based on feature correlations to detect which features might have impacted our evaluation. Our model could be applied to distributed systems of several kind as far as they respect few properties. Future work could be lead on the unsupervised way to construct features in order to automate the analysis process. Second, we specifically applied this framework to the self-assessment and self analysis of dynamic wireless networks in the environment of Ns3. After having defined an evaluation policy for our network, we provided a distributed algorithm derived from existing average consensus schemes to compute this assessment. We evaluated this algorithm under various networking conditions to describe its sensitivity to load and topology dynamicity. To improve the algorithm accuracy, future work will be pursued on the use of the designed feature to have a better estimation propagation.

Acknowledgment

This work is partially funded by the French National Research Agency (ANR) under the project ANR VERSO RESCUE (ANR-10-VERS-003)

References

[1] “Cisco visual networking index: Global mobile data traffic forecast update, 2012-2017,” White Paper, Cisco, February 2013.

[2] “Substitution networks based on software defined networking,” in *Ad Hoc Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, J. Zheng, N. Mitton, J. Li, and P. Lorenz, Eds., 2013, vol. 111.

[3] The AT&T Labs Research website. [Online]. Available: http://www.research.att.com/projects

[4] E. Bloedorn, A. D. Christiansen, W. Hill, C. Skorupka, L. M. Talbot, and J. Tivel, “Data mining for network intrusion detection: How to get started,” Tech. Rep., 2001.

[5] P. Casas, J. Mazel, and P. Owezarski, “Unsupervised network intrusion detection systems: Detecting the unknown without knowledge,” *Comput. Commun.*, vol. 35, no. 7, pp. 772–783, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2012.01.016

[6] M. Arlitt and C. Williamson, “An analysis of tcp reset behaviour on the internet,” *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 37–44, 2005.

[7] B. Rengarajan, “Data mining and coordination to avoid interference in wireless networks.”

[8] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J.-H. Hong, and A. Dey, “Factors influencing quality of experience of commonly used mobile applications,” *Communications Magazine, IEEE*, vol. 50, no. 4, pp. 48–56, 2012.

[9] K. Topley and V. Krishnamurthy, “Average-consensus in a deterministic framework ;part i: Strong connectivity,” *Signal Processing, IEEE Transactions on*, vol. 60, no. 12, pp. 6590–6603, 2012.

[10] L. Xiao, “A scheme for robust distributed sensor fusion based on average consensus,” in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN*, 2005, pp. 63–70.

[11] L. Xiao, S. Boyd, and S. jean Kim, “Distributed average consensus with least-mean-square deviation,” *Journal of Parallel and Distributed Computing*, vol. 67, pp. 33–46, 2005.