Wi-Fi Throughput Estimation and Forecasting for Vehicle-to-Infrastructure Communication

Daniel Teixeira[a,b,*], Rui Meireles[c], Ana Aguiar[a]

[a]*Instituto de Telecomunicações, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, 4200–465 Porto, Portugal*
[b]*Cisco Systems Inc., Lisbon, Portugal*
[c]*Computer Science Department, Vassar College, USA*

Abstract

Vehicles increasingly need to connect to external networking infrastructure, to support applications such as over-the-air updates, edge computing, and even autonomous driving. The ubiquity of IEEE 802.11 Wi-Fi makes it ideal for opportunistic vehicular access. However, that ubiquity also creates a problem of choice. In a heterogeneous Wi-Fi environment, where different networks coexist, it becomes important for vehicles to be able to pick the best-performing one. Focusing on delay-insensitive traffic, we equate network performance with throughput. To inform network selection we aim to first estimate current throughput, and then forecast its evolution through time. In order to avoid introducing load onto the network, we estimate throughput using only passively observable variables such as signal strength. We used Symbolic Regression (SR) and an Unscented Kalman Filter (UKF) to develop a computationally inexpensive estimation model — UKF-SR. We trained and tested this model using experimental data featuring 802.11n, ac, and ad networks. UKF-SR proved competitive against more expensive models such as shallow neural networks. To predict future throughput, we explored both general time-series forecasting models such as Autoregressive Integrated Moving Average (ARIMA), and domain-specific ones based on mobility information. The latter clusters historical throughput according to attributes such as vehicle position and direction of movement, using the cluster's average as the forecast. An evaluation using experimental data showed the mobility-based models to meaningfully outperform general forecasting.

Keywords: Vehicular Wi-Fi, symbolic regression, vehicular offloading, throughput estimation, throughput forecasting.

1. Introduction

Vehicle-to-Infrastructure (V2I) communication is increasingly important. It enables myriad applications that promote road safety, efficiency, and entertainment. For example, Tesla vehicles upload information they collect to the cloud to help develop autonomous driving algorithms [1]. And they also download media for streaming, traffic information, and software updates for the vehicles' subsystems.

Such traffic is typically sent over a cellular connection. But, with ever increasing capacity demands — Cisco reports compound annual traffic growth rates between 7 and 30 % [2] — there is a need for alternatives. The availability of IEEE 802.11 Wi-Fi Access Points (APs) usable from roadways, particularly in urban environments, makes them a compelling proposition. A large scale wardriving study conducted in the city of Porto, Portugal [3] found over 80 thousand APs visible from roads. Roughly 28 % of them were open, i.e., unsecured, and thus good candidates for data offloading. The large majority ($\sim$90 %) of open APs required post-connection authentication, e.g., through a captive portal. However, they were all controlled by a very small number of providers, such as Fon Wi-Fi [4], that many consumers already subscribe to. Additionally, authentication could be automated, and carried over across APs of the same provider.

The abundance of Wi-Fi networks poses an opportunity but also a challenge, as vehicles must choose which one to use at any given moment. First, different networks may use different IEEE 802.11 standards, which strike different balances between bandwidth and coverage range. Second, the high degree of mobility associated with vehicular environments limits the amount of time during which each network can be used.

One reasonable goal, particularly for delay-insensitive traffic, is to pick the network that maximizes the amount of data that can be offloaded [5, 6], which equates to cumulative throughput over time. Determining this for a given network can be decomposed into two subproblems:

[⋆]This work is a result of project FLOYD (POCI-01-0247-FEDER-045912), funded by the European Regional Development Fund (FEDER), through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) and by Portuguese National Funds (OE), through Fundação para a Ciência e Tecnologia, I.P.; and UIDB/50008/2020, funded by the applicable financial framework (FCT/MCTES) (PIDDAC).

[*]Corresponding author.

Email addresses: danielfilipeteixeira24@gmail.com (Daniel Teixeira), rui.meireles@vassar.edu (Rui Meireles), anaa@fe.up.pt (Ana Aguiar)

1. **Estimate** the throughput that the network can currently provide.

2. **Forecast** how said throughput will evolve in the near future, given the current throughput estimate and other variables of interest.

Throughput estimation is often done by actively injecting probe traffic into the network [6, 7], requiring an active connection and introducing unwanted congestion. In contrast, we explore the possibility of leveraging passively-observable variables such as distance to the AP and received signal strength. Moreover, we aim to create an easily-explainable model that is also computationally inexpensive and thus suitable for embedded devices.

With these goals in mind our approach was to take realistic experimental Wi-Fi V2I communication performance data, and apply Symbolic Regression (SR) offline to obtain a simple throughput estimation model. We then refined the model by incorporating an Unscented Kalman Filter (UKF), yielding a variant we call UKF-SR.

To assess its generalization capabilities we evaluated SR-UKF against standard machine learning (ML) models such as Decision Trees (DT) and Shallow Neural Networks (SNN), on a separate testing dataset. Despite its simplicity, SR-UKF outperformed the competition.

Switching Wi-Fi networks involves a certain amount of downtime. Therefore, once current throughput has been estimated, it's important to forecast how it will evolve over time. This will help determine whether it is worth switching to a different network.

The simplest way to do this is to consider historical throughput estimates as a time series and apply a standard forecasting algorithm such as a simple moving average or Autoregressive Integrated Moving Average (ARIMA). However, this strategy ignores external variables that may play an important role in the forecast.

For vehicular networks in particular, mobility information such as vehicle location and direction of movement are key connectivity predictors, as they indirectly encode all factors affecting signal attenuation, such as distance, line-of-sight conditions, and obstacles causing multipath interference. Thus, we consider the MRDP (Mobility, Road, Direction, Position) algorithms introduced by Meireles et al. [6]. These algorithms forecast throughput to be the average of previously-estimated throughput values for a given set of mobility conditions. When evaluated on our experimental dataset, they substantially outperformed traditional time-series forecasting algorithms.

In summary, we make the following contributions:

1. Describe how we collected network performance data in a realistic Wi-Fi-diverse vehicular scenario to test our estimation and forecasting models (§3).

2. Present our UKF-SR symbolic regression throughput estimation model, and evaluate it against standard ML models from the literature using separate training and testing datasets (§4).

3. Describe and thoroughly evaluate both general and domain-specific throughput forecasting models, in isolation and in conjunction with our estimation model (§5).

In summary, by exploring throughput estimation and forecasting solutions, this work paves the way for a practical dynamic network selection system for V2I communication. It extends a previous conference article on throughput estimation [8], adding the contribution on forecasting.

2. Related work

We consider two types of related work. That concerning estimation of current throughput, and that concerning forecasting of future throughput.

Estimating throughput ordinarily implies injecting large amounts of traffic into the network, rendering it unusable from congestion. Li et al. [9] proposed an alternative packet-pair-based algorithm, which requires only a small number of probes. However, the algorithm assumes that network conditions are stable, which is not true in vehicular environments. Furthermore, to generate probes a user must be connected to an AP, while in the context of network selection the throughput estimate is needed prior to connection establishment.

The alternative is to rely on passive measurements alone, thus not generating additional traffic. A seminal work in throughput estimation is that of Mathis et al. [10], where the authors introduced a mathematical TCP throughput model, function of packet loss probability. However, it does not apply to wireless environments, where losses do not necessarily imply congestion.

Samba et al. [11] performed throughput estimation in cellular Long Term Evolution (LTE) networks by leveraging several data sources, such as context information (e.g., distance and speed) and physical layer measurements (e.g., signal strength). Using a random forest algorithm the authors obtained a model capable of explaining 84 % of the variation in throughput.

In this work we introduce a throughput estimation model that employs only client-side passive measurements, accessible prior to establishing a connection, and trained specifically for the vehicular Wi-Fi access use case.

Consider now the problem of forecasting future throughput. Liu & Lee [7] surveyed existing solutions and categorized them as: (i) formula-based, using mathematical expressions; (ii) history-based, predicting throughput using past measurements; or (iii) machine learning model-based.

He et al. [12] presented a formula-based method to forecast TCP throughput, considering maximum segment and window sizes, round-trip time, and packet loss rate. In this study, the authors also explored history-based prediction with moving average and non-seasonal Holt-Winters algorithms. The history-based methods performed better.

However, they depend on the availability of prior measurements, unlike the formula-based ones.

Wei et al. [13] developed a history and ML-based approach to TCP throughput forecasting in mobile networks named TRUST. It first classifies the user's mobility pattern. The classification is then used to pick a Long Short-Term Memory (LSTM) neural network model to predict throughput. The authors showed it outperformed moving average and hidden Markov models across multiple different mobility scenarios, such as walking or riding a train.

In our work we focus on the quality of the underlying channel rather than how the transport protocol's flow and congestion controls interact with it. For this reason, we use UDP. Further, we focus specifically on opportunist vehicular Wi-Fi access, which is characterized by very fast mobility. The work closest to ours is that of Meireles et al. [6], who introduced a mobility clustering-based family of algorithms that forecast throughput as the average of previously-observed throughput values for a given set of mobility variables – more in §5.2. However, these algorithms were evaluated using actively-measured throughput values, rather than passive estimates. Here we refine the algorithms and evaluate how well they perform with passive estimates.

3. Datasets

To develop a good throughput estimation model, it is essential to use data collected in a realistic environment. We had access to such a dataset, created in the context of related prior work [6]. We shall refer to it as the Gaia dataset, in reference to where it was collected.

We decided to complement it with a new dataset, the Porto dataset, collected at a different time and location. This let us use different train and test datasets for our throughput estimation model, thus assessing how well it generalizes across different environments. In this section we describe both datasets, highlighting their differences.

3.1. Porto dataset

3.1.1. Experimental setup

We drove a vehicle around a stationary access point, while transmitting data through three different networks, each using a different Wi-Fi standard — IEEE 802.11n, ac, and ad, in parallel.

Fig. 1a depicts the location where the experiments took place — Rua D. Frei Vicente da Soledade e Castro in Porto, Portugal. This street runs roughly east-west, and is lined by trees on both sides. It also sits lower than the surrounding terrain, by roughly 2 m. Weather was cloudy but dry, with a temperature of around 25 °C.

The AP remained stationary at the location shown, while the client vehicle did a total of 10 laps around the indicated circuit, traveling counterclockwise. Speed varied between 0 and 40 km/h. Vehicular traffic was substantial, which added to the environment's multipath and line of sight obstructions.

Fig. 2 illustrates the vehicles and hardware devices used in the experiments. We placed a TP-Link Talon AD7200 [15] router on the roof of each vehicle. These routers feature 802.11n, ac, and ad interfaces, making them ideal for our purposes. Given that the original firmware is not user-programmable, we installed LEDE-AD7200, a modified version of the Linux-based LEDE operating system specifically created for the AD7200 routers by the Talon Tools project [16].

Taking advantage of the programmable environment, we wrote simple data producing and consuming applications and deployed them on the routers themselves. Pseudo-random data was sent over the three radio interfaces, from the mobile client to the static AP. Tab. 1 shows the channels used, all of which are independent.

Standard	Channel #	Center freq. (GHz)	Bandwidth (MHz)
802.11n	6	2.437	20
801.11ac	40	5.2	40
802.11ad	2	60.48	2160

Table 1: Wi-Fi channels used for the experiments.

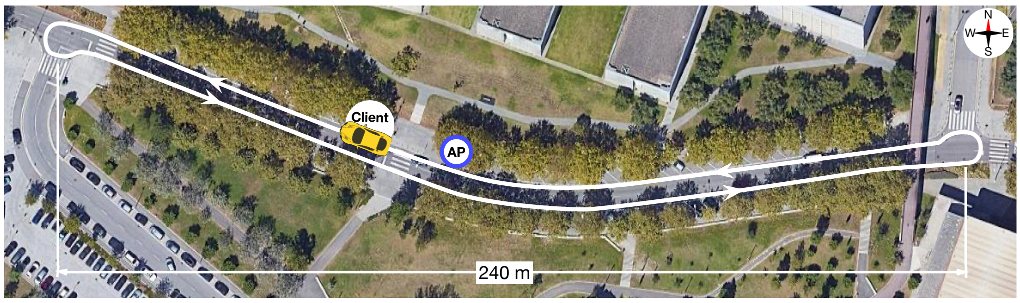
Throughput was measured by the receiving AP and sent as feedback to the client over the 802.11n network, as it featured the longest range. This information was saved on a database, which, due to memory constraints, ran on an external device connected to the client router through Gigabit Ethernet.

Location and time information for the mobile client was provided by a MikroElektronika GNSS 5 click receiver [17], which features a 10 Hz GPS module. However, to increase the number of packets used to compute each throughput data point, our entire analysis used a granularity of 1 Hz. This means we downsampled the GPS information, using the first sample in each second to represent that entire second. Due to a lack of additional GNSS 5 devices, the AP was equipped with a commodity 1 Hz GPS receiver.

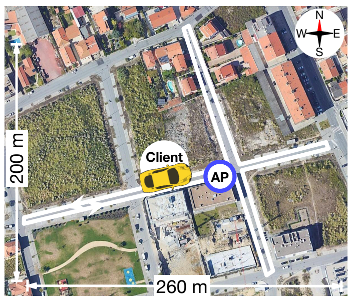
3.1.2. Collected data

Using the described setup, we collected a dataset with over 3750 samples (1250 per Wi-Fi standard). Each sample pertains to a specific (timestamp, network) combination. The timestamp resolution is 1 Hz. A sample contains the mobile client's location coordinates, heading, and speed, along with the measured signal quality, throughput, and transmission data rate. The data can be downloaded from Zenodo [18].

Fig. 3 summarizes the collected data through some of its metrics. Namely, measured throughput, vehicle speed, and GPS horizontal dilution of precision (HDOP). Isik et al. [19] suggest the use of dilution of precision values to validate the integrity of GPS estimates. Values below 1 are considered "ideal", and below 2, "excellent". Following these guidelines, we consider the collected GPS data trustworthy, as HDOP values were consistently below 1.



(a) Porto mobility pattern. Access point GPS coordinates: 41.177, -8.59585.



(b) Gaia data. GPS: 41.112, -8.631.

Figure 1: Experiment locations and client mobility patterns used in the Porto and Gaia datasets. Imagery ©2022 CNES / Airbus, IGP/DGRF, Maxar Technologies, Map data ©2022 Google.

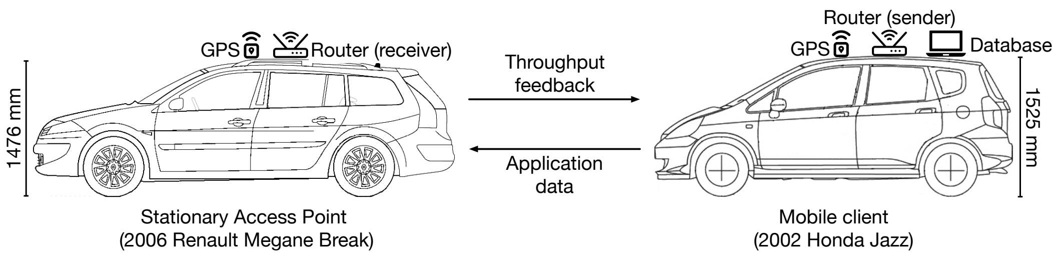


Figure 2: Porto experimental setup. Vehicles are drawn to scale. Blueprints courtesy of getoutlines.com [14].

The mobile client's speed ranged between 0 and 40 km/h, with average values in the range of 5 to 30 km/h. The lowest speed values were observed at both extremities, where we had to slow down to perform a U-turn, and at the initial/final position, where the vehicle remained stationary for a period of time at the experiments' beginning and end.

Throughput varied significantly across the different networks, with a maximum average throughput of 50 Mbps for 802.11n, 150 Mbps for 802.11ac, and 200 Mbps for 802.11ad.

The 802.11n and ac networks remained connected throughout the experiments, even at distances of over 125 m. On the other hand, 802.11ad was only able to communicate at distances of less than 25 m. This is due to its use of 60 GHz frequencies, which suffer from much greater attenuation than the 2.4 and 5.2 GHz used by 802.11n and ac, respectively.

802.11n's throughput did not vary significantly for distances above 25 m. 802.11ac's throughput on the other hand, initially decreased with distance, but then increased as the vehicle scrubbed speed near the turnaround points — from 25 to 75 Mbps. This behavior reveals an inverse relationship between speed and throughput, suggesting that 802.11ac is more sensitive to mobility than 802.11n.

Signal strength was similar for both 802.11n and ac, peaking at −30 dBm and dropping to between −60 and −70 dBm as the vehicle moved away from the AP. Like throughput, 802.11ad's signal strength was only reported for very short distances.

3.2. Gaia dataset

Just like the Porto dataset, the Gaia dataset contains network performance data collected in a V2I communication scenario where a vehicle drove a circuit around an access point. The client exchanged data with the AP over three different networks, one IEEE 802.11n, one ac, and one ad, in parallel. This dataset contains essentially the same time-indexed mobility and throughput information as the Porto dataset, with the same 1 Hz resolution.

The experimental setup used was also very similar to the one used to collect the Porto dataset. There are a total of five main differences between the two datasets: (i) location, (ii) mobility pattern, (iii) hardware, (iv) traffic flow direction, and (v) number of active clients.

As per Fig. 1a, in the Porto dataset the client drove back and forth along a tree-lined road. In the Gaia dataset, the AP remained at the corner of a suburban intersection as the client approached and left the intersection in every possible direction combination. Fig. 1b depicts the environment and a simplified version of this mobility pattern.

In terms of hardware, the same Talon AD7200 routers were used for IEEE 802.11ad communication, but the devices used for 802.11n and ac differed. The devices were also mounted on the roof of two vehicles (although of different makes and models, sizes and shapes).

In the Porto dataset, data flowed from client to AP. In the Gaia dataset, it flowed in the reverse direction. However, we believe it reasonable to assume channel symmetry, as both AP and client used the same exact communication devices, with the same transmission power and antennas.

Finally, due to logistical reasons, the Porto dataset featured a single active client per network. In the Gaia

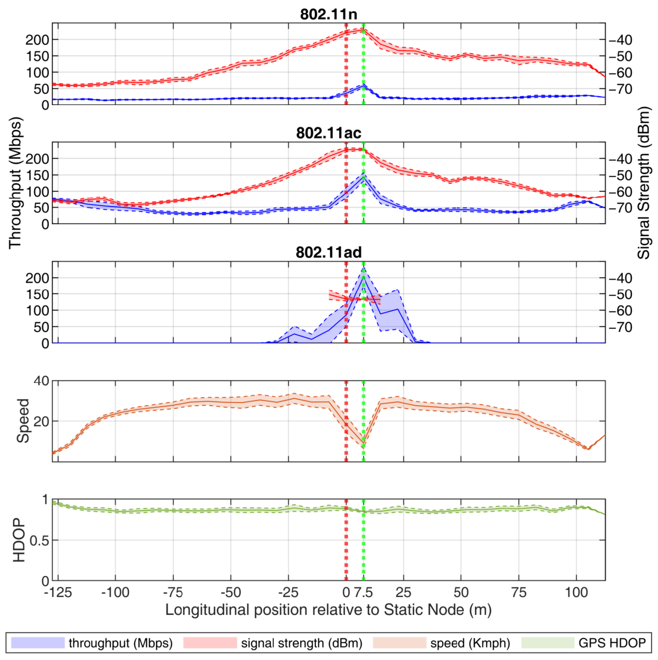


Figure 3: Mean throughput, RSSI, speed, and GPS horizontal dilution of precision as a function of the client's longitudinal position relative to the AP (negative for West and positive for East). The red vertical dotted line denotes the AP's position; the green vertical dotted line indicates the client's initial (and final) position. Ranges represent the 95 % confidence intervals.

dataset the experiments were first performed with a single client per network and then repeated with two and three active clients for 802.11n and ac. The goal being to capture the impact of the number of active clients on throughput.

This Gaia dataset was described in detail in an earlier work [6]. It is also available for download on Zenodo [20].

4. Throughput estimation

This section presents and evaluates a lightweight symbolic regression-based model to estimate current throughput without active measurements.

4.1. Symbolic regression-based estimation

Symbolic Regression (SR) is a data-driven technique used to uncover mathematical expressions that can be used to model a certain feature from a dataset [21]. The mathematical expressions produced can be sanity-checked against domain-specific knowledge (e.g., we expect throughout to be positively correlated with signal strength), and are computationally inexpensive to apply. In this section we leverage SR to develop throughput estimation models based on a Wi-Fi vehicular dataset for IEEE 802.11n/ac/ad access networks.

4.1.1. Methodology

Symbolic regression is not a statistical method. Instead, it implies searching the parameter space to find the combination yielding the best fit. For this purpose we used GPTIPS2 [21]. GPTIPS2 is a genetic-programming-based algorithm that searches for linear combinations of mathematical functions and input variables. It formulates an equation similar to a linear model, but with potentially non-linear terms. These terms are represented by tree structures, where leaves are variables and constants, and intermediate nodes are mathematical operators, e.g., plus, minus, cosine, etc.

To develop the estimation models, we followed the pipeline in Fig. 4, where models go through training, validation, and testing phases. We used the Gaia dataset (§3.2) for training and validation, applying a random 70-30 % split, and the Porto dataset (§3.1) for testing. Using two different datasets let us analyze how well the models generalize across environments. To ensure that the models are kept simple and do not over-fit the data, we limited their structure to six terms, each represented by a tree with maximum depth of four. The algorithm was executed 25 times and in each run it evolved 300 models through 100 generations, resulting in a set of 7500 trained models. At the end of each generation, the algorithm runs a tournament selection with a size of 30, to select the best models for crossover. To choose a model, we plotted the Pareto front with regards to the models' goodness-of-fit, based on the validation set and their expressional complexity. The expressional complexity is calculated as the sum of the number of nodes of all sub-trees [22]. The chosen model is the one among those in the Pareto front exhibiting the best empirical trade-off between performance and complexity.

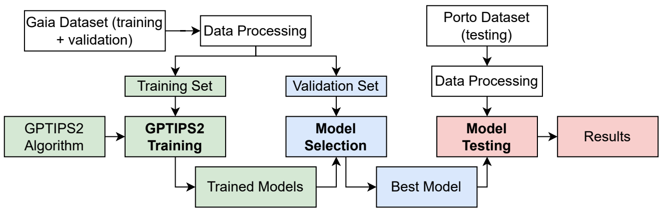


Figure 4: Symbolic regression pipeline methodology for training, validating, and testing the discovered models.

4.1.2. Resulting estimation models

We performed SR for each Wi-Fi standard separately, following the previously described methodology. The resulting estimation models for $tput_n$, $tput_{ac}$, and $tput_{ad}$ are described by the following equations:

$$\begin{aligned}\hat{tput}_n &= 0.7111 * RSSI - 2.479 * N_{users} \\ &\quad + 11.88 * e^{-N_{users}} + 62.02\end{aligned} \tag{1}$$

$$\begin{aligned}\hat{tput}_{ac} &= 1.409 * RSSI - 2.667 * v \\ &\quad + 44311 * \cos(\frac{N_{users}}{RSSI}) \\ &\quad - 11.14 * \cos(N_{users}) - 36.77 * d^{\frac{1}{4}} - 44022\end{aligned} \tag{2}$$

$$\hat{tput}_{ad} = 67.75 * N_{retries}{}^{\frac{1}{4}} - 88.18 * \tanh(v) - 6.348 * \sqrt{N_{retries}} - 75.37 * e^{-(v+N_{beacons})^3} + 88.83 \tag{3}$$

where $\hat{tput}_n$, $\hat{tput}_{ac}$, and $\hat{tput}_{ad}$ are the throughput estimates in Mbps, $RSSI$ is the received signal strength in dBm, d is the vehicle-AP distance in meters, v is the vehicle's speed in m/s, N_{users} is the number of users connected to the AP, and $N_{beacons}$ and $N_{retries}$ are the number of beacon frames received and retransmissions in the last second, respectively.

The training and validation datasets use RSSI to represent signal quality for 802.11n and ac, but Signal-to-Noise Ratio (SNR) for 802.11ad. The testing dataset uses RSSI for all standards. Therefore, if we were to train a model with SNR we would be unable to test it. This is why, unlike the other equations, the one for $\hat{tput}_{ad}$ does not take signal quality into account. Instead, it uses the number of beacons and re-transmissions as proxies. The use of these variables in place of RSSI or SNR may lead to poor performance. To circumvent this limitation and improve the model's accuracy, we generated synthetic RSSI values for 802.11ad using a simplified log-distance path loss model:

$$RSSI = RSSI_0 - 10\,\gamma\,\log_{10}\left(\frac{d}{d_0}\right) + X \qquad X \sim \mathcal{N}(0, \sigma) \tag{4}$$

where d_0 is the reference distance (i.e., one meter), $RSSI_0$ is the RSSI at d_0, γ is the path loss exponent, and X is a zero-mean Gaussian random variable with σ standard deviation, reflecting the effect of fading. The path loss exponent and fading standard deviation were set to $\gamma = 1.59$ and $\sigma = 2.1$, respectively, following the results of Karedal et al. for suburban environments [23]. $RSSI_0$ was measured to be -53 dBm for our TP-Link Talon AD7200 access points.

The proportion of 802.11ad samples from disconnected periods is also concerning, since, given ad's short range, they correspond to the vast majority of the dataset. The large amount of zero-throughput samples influences the model's training, forcing it to not only learn how to estimate throughput, but also to classify periods of connection versus disconnection (i.e., zero throughput versus non-zero throughput). However, we can easily determine when the vehicle has a viable connection with an AP, and the model should not focus on any other task apart from estimating throughput. Therefore, we reran the SR algorithm with the generated RSSI values and trained using only non-zero throughput samples. Throughput for 802.11ad can then be estimated as:

$$\hat{tput}_{ad} = 0.7334 * RSSI + 47.74 * \sin(v * RSSI) - 112.6 * \tanh(v)^{1/4} - 115.8 * \tanh(\cos(v)) * \log(N_{users})^2 + 387.9. \tag{5}$$

4.2. Estimation refinement with Kalman filters

When estimating throughput with ML or SR models, or manually-discovered formulas, we are subject to two primary sources of noise: (i) process noise, due to inaccuracies in the fitting of the data; and (ii) measurement noise, resulting from inherent errors in the model's inputs (e.g., RSSI or distance). In this section we improve the SR model's performance by minimizing errors using recursive Bayes filters.

4.2.1. Recursive Bayes filters

Recursive Bayes filters combine a theoretical model of how a state estimate (e.g., vehicle's speed and direction) evolves over time, with an observational model that defines what measurements we should expect given an estimated state. The most popular recursive Bayes filter is the Kalman Filter (KF), an optimal state-estimator for linear systems with additive Gaussian noise. Intuitively, a KF applies a weighted average over the estimated state and the expected state from the measurements, where the weight is defined as the Kalman gain.

Our goal in using a KF is to build a state-space model that leverages the vehicle's distance, direction, and speed to estimate throughput using the SR model. But, if we include Eqs. (1), (2), and (5) in a state-space model we will have a non-linear system, breaking KF's linearity assumption. The Extended Kalman Filter (EKF) is a non-linear version of the Kalman Filter, where a first-order Taylor series approximation is used to linearize the system. Still, the use of a first-order approximation may lead to the filter diverging from an acceptable estimate.

An alternative to EKF has been proposed by Julier & Uhlmann [24]. It uses an unscented transformation to linearize the system with a small set of chosen points. The Unscented Kalman Filter (UKF) is equivalent to a third-order Taylor-series expansion, severely reducing the divergence probability relative to the EKF algorithm.

The UKF algorithm consists of prediction and correction steps. In the first, a state-space model is used to predict the state for time t, defined by the mean x_t^- n-dimensional vector and the covariance matrix P_t^-, based on a set of *sigma* points that characterize the state from $t-1$. This can be expressed mathematically as:

$$x_t^- = \sum_{i=0}^{2n} w_i^{(m)}\, f(\mathcal{X}_i) \qquad P_t^- = \sum_{i=0}^{2n} w_i^{(c)}\,(f(\mathcal{X}_i) - x_t^-)\,(f(\mathcal{X}_i) - x_t^-)^T + Q_t \tag{6}$$

where $\mathcal{X}$ is a set of $2n$ *sigma* points, $w^{(m)}$ and $w^{(c)}$ are the mean and covariance weights, f is the state-space model, and Q_t is the process noise matrix. In some systems we may extend the formulation to include control inputs. The *sigma* points $\mathcal{X}$ and their weights represent the non-linear distribution for the state in time $t-1$ (see [24]).

In the correction step, UKF uses the set of measurements for time t and the measurement model to correct the predicted state. The corrected state, described by the mean vector x_t and covariance matrix P_t, is given as:

$$\begin{aligned}
\mathcal{Z}_t &= \sum_{i=0}^{2n} w_i^{(m)}\, h(\mathcal{X}_i) \\
S_t &= \sum_{i=0}^{2n} w_i^{(c)}\, (h(\mathcal{X}_i) - \mathcal{Z}_t)\, (h(\mathcal{X}_i) - \mathcal{Z}_t)^T + R_t \\
T_t &= \sum_{i=0}^{2n} w_i^{(c)}\, (\mathcal{X}_i - x_t^-)\, (\mathcal{X}_i - x_t^-)^T \\
K_t &= T_t\, S_t^{-1} \\
x_t &= x_t^- + K_t\, (z_t - \mathcal{Z}_t) \\
P_t &= (I - K_t\, T_t)\, P_t^-
\end{aligned} \tag{7}$$

where $\mathcal{Z}_t$ and S_t are the mean and covariance of the expected measurement's distribution given $\mathcal{X}$ *sigma* points, h is the measurements model, R_t is the measurement noise matrix, T_t is the covariance matrix between the *sigma* points and the predicted state, z_t is the measurements' vector, and K_t is the Kalman gain.

4.2.2. Filtered estimation models

To model our system using UKF, our state must represent the set of input features for the SR models in Eqs. (1), (2), and (5), as well as other available attributes associated with said features. Considering that the testing dataset was collected with a single connected user, the SR models used in the UKF are simplified versions where $N_{users} = 1$. That said, our state vector x_t is defined as:

$$x_t = \left[lon,\, lat,\, v,\, \theta,\, d,\, RSSI,\, tput\right] \tag{8}$$

where lon and lat are geographical coordinates, and θ is the vehicle's direction. The corresponding state-space model f is:

$$f(x_{t-1}) = \begin{cases} lon_t & = lon_{t-1} + \Delta_{lon} * (180/\pi) \\ lat_t & = lat_{t-1} + \Delta_{lat} * (180/\pi) \\ v_t & = v_{t-1} \\ \theta_t & = \theta_{t-1} \\ d_t & = \text{haversine}(lon_t,\, lat_t) \\ RSSI_t & = f_{RSSI}(d_t,\, v_t) \\ tput_t & = f_{tput}(d_t,\, v_t,\, RSSI_t) \end{cases} \tag{9}$$

where

$$\Delta_{lon} = \frac{\cos(\theta_{t-1}) * v_{t-1}}{R * \cos(lat_{t-1} * \frac{\pi}{180})},\ \Delta_{lat} = \frac{\sin(\theta_{t-1}) * v_{t-1}}{R}$$

, with R being the earth's radius, approximated as 6371 km.

In Eq. (9), throughput, represented by f_{tput}, is given by the simplified (i.e., $N_{users} = 1$) SR models.

Speed and direction of movement are assumed constant, and used to predict the vehicle's current position from the previous one. The distance between the vehicle and AP positions is computed using the Haversine formula.

RSSI is given by f_{RSSI}, which predicts the current RSSI based on the predicted speed and distance. For 802.11ad this function corresponds to the log-distance path loss model defined in Eq. (4), used to generate synthetic RSSI values. For 802.11n and ac, we used the previously described SR methodology to create two prediction models that dictate how distance and speed influence RSSI:

$$\begin{aligned} RSSI_n = 0.050 * v - 2.739 * \log(d^3) \\ - 1.090 * \sqrt{v + d} - 23.22 \end{aligned} \tag{10}$$

$$RSSI_{ac} = -16.24 - 12.43 * \log(d). \tag{11}$$

To analyze these models' performance, we compared them against the log-distance path loss model of Eq. (4). The Root-Mean-Squared Errors (RMSE) for the log-distance path loss model were 32.46 and 25.80 dBm for 802.11n and ac, respectively, but only 7.01 and 8.44 dBm for the SR models.

The measurement model h was implemented as an identity function, i.e., measured values were kept unchanged. The measurements vector z_t has the same structure as x_t in Eq. (8).

4.2.3. Noise matrices

UKF is regulated by a process-noise matrix Q_t and a measurement-noise matrix R_t, which define the variance of state and measurements, respectively. Incorrectly defining them may result in poor performance and even divergence. While there are multiple ways to estimate noise matrices, we followed a simple approach, relying on domain knowledge and manual tuning. For simplicity, we expressed Q_t and R_t as diagonal matrices. This approach will yield sub-optimal performance, but serves as a quick way to implement the system and demonstrate its potential.

The manual tuning of the noise matrices was conducted based on the testing dataset, which introduces biases when measuring the model's accuracy. However, we expect the introduced bias to be negligible since our approach will naturally lead to a poorer choice of matrices compared to more statistically-advanced approaches, such as adaptive filters or the Autocovariance Least-Squares method proposed by Odelson et al. [25].

4.3. Evaluation

In this section we compare our SR and UKF-SR models' performance against that of classic ML methods, using the Porto dataset, described in §3.1.

4.3.1. SR and UKF-SR models

Fig. 5 compares the actual throughput measurements with estimated throughput using the SR and UKF-SR models. Both uncorrected (x_t^- from Eq. (6)) and corrected (x_t from Eq. (7)) predictions are shown.

Results indicate that UKF-SR's pre-corrected estimates are less accurate than those made by plain SR, due

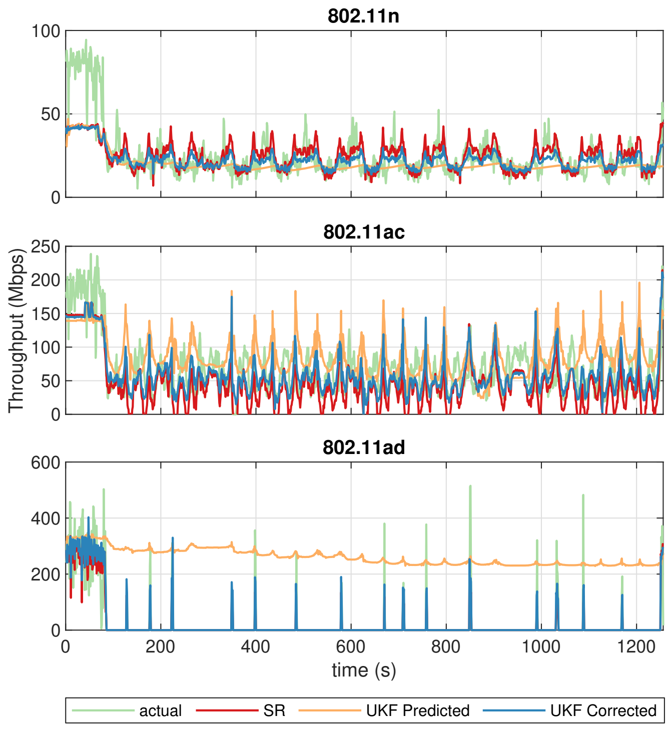


Figure 5: Estimated throughput using SR and UKF-SR models; "UKF Predicted" corresponds to x_t^- in Eq. (6), and "UKF Corrected" to x_t in Eq. (7).

to the simple state-space model used and its dependence on values from time $t-1$. However, this is thoroughly compensated by the correction step. In the end, by leveraging Kalman filtering, the RMSE relative to plain SR was decreased by 8.33 % for 802.11n, 9.85 % for 802.11ac, and 13.94 % for 802.11ad.

4.3.2. Machine learning comparison

To put our models' performance into perspective, we benchmarked them against commonly available ML models. The chosen models were Multiple Linear Regression (MLR), Support-Vector Regression (SVR), Decision Trees (DT), Random Forests (RF), and Shallow Neural Networks (SNN). They were trained following a similar approach to that used for SR, where models go through separate training, validation, and testing phases. But, while for SR the validation step consisted on manually choosing a model among those in the Pareto front, for ML we used Bayesian optimization to maximize the models' performance by tuning their hyperparameters. These vary between models, e.g., the number of neurons in a layer for SNN, and the regularization value for SVR. MLR is an exception, as it features no hyperparameters.

Fig. 6 shows the empirical cumulative distribution function (ECDF) plots for the models' absolute errors. For 802.11n, the ECDF error curves are similar among all models. SR, depicted in orange, managed to compete with Gaussian SVR ML, and surpassed SNN, RF, and DT up to the 60th percentile. UKF-SR, depicted in red, was the best-performing model, improving upon the SR estimates up to roughly the 92nd percentile.

For 802.11ac, two distinct groups can be observed: a better-performing group consisting of SR, UKF-SR, RF, DT, and Gaussian SVR models; and a worse-performing group composed of Linear SVR, MLR, and SNN.

For 802.11ad, due to the large proportion of zero-throughput samples from disconnected periods, around 90 % of predictions were error-free across the board. Thus, we focus on the remaning 10 % of estimates, pertaining to connected-period samples. Performance across models varied less than for 802.11ac, with the SR model showing good results, but being surpassed by SNN, DT, and Gaussian SVR by a small margin. However, despite being based on SR, UKF-SR competed with the more computationally demanding algorithms quite well.

Tab. 2 shows the different models' RMSE in both absolute form, and as a percentage of a UKF-SR baseline. UKF-SR was the most accurate model for 802.11n and ac, and the second-best for 802.11ad, with RMSE values of 11.53, 28.43, and 39.11 Mbps, respectively. For 802.11n, DT was second-best, with an error 4.94 % higher than

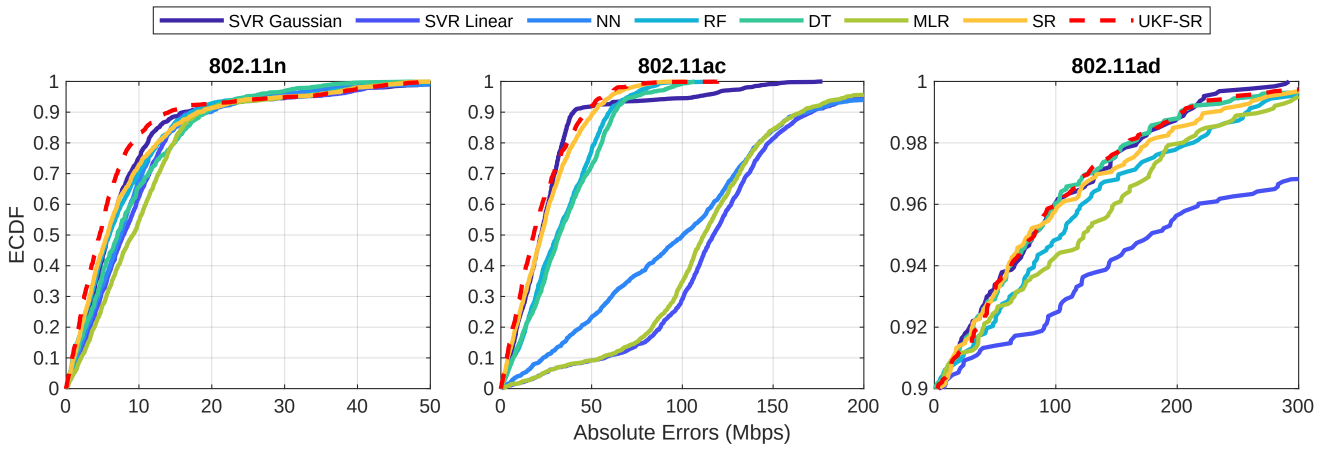


Figure 6: ECDF plots of the absolute out-sample errors for the ML, SR and UKF-SR models.

Models	**802.11n**		**802.11ac**		**802.11ad**	
	RMSE	%	RMSE	%	RMSE	%
MLR	13.77	(+19.43)	124.94	(+339.47)	51.90	(+32.70)
RF	12.46	(+08.07)	39.26	(+38.09)	50.92	(+30.20)
SNN	13.07	(+13.36)	129.02	(+353.82)	42.41	(+08.44)
DT	**12.10**	(+04.94)	41.77	(+46.92)	42.41	(+08.44)
SVR Gauss	12.76	(+10.67)	39.62	(+39.36)	**38.70**	(-01.05)
SVR Linear	13.40	(+16.22)	130.38	(+358.60)	116.09	(+196.83)
SVR 2D	14.42	(+25.07)	39.74	(+39.78)	—	—
SVR 3D	21.29	(+84.65)	39.95	(+40.52)	40.92	(+04.63)
SR	12.49	(+08.33)	**31.23**	(+09.85)	44.56	(+13.94)
UKF-SR	**11.53**	—	**28.43**	—	**39.11**	—

Table 2: Performance results of the ML, SR and UKF-SR models for 802.11n/ac/ad, considering RMSE.

UKF-SR, followed by the RF model with 8.07 %. SR was the second-best model for 802.11ac, with an error 9.85 % higher than UKF-SR. The third-best, RF, was significantly worse, with an error 38.09 % higher than UKF-SR. For 802.11ad, UKF-SR was outperformed by the Gaussian SVR model, although by a very small margin of 1.06 %. The third-best performing model for 802.11ad was SVR 2D, with an error 4.63 % higher than Gaussian SVR.

5. Throughput forecasting

This section presents and evaluates different approaches to throughput forecasting, a new contribution relative to the preceeding conference article [8].

5.1. Time-series-based forecasting

This subsection considers throughput as a time series — a sequence of data points ordered by time — and explores common time-series forecasting techniques [26].

5.1.1. Basic moving average

A moving average forecasts a variable at time t, y_t, as the average of variable values observed in a window of time prior to t. It can be written as: $y_t = \sum_{i=i}^{n} \beta_i y_{t-i}$, where n is the number of prior variable samples, or lags, to be included, and β_i the weight associated with the $t-i$ lag. The weights add up to one, i.e., $\sum_{i=i}^{n} \beta_i = 1$. Common variants include:

Arithmetic moving average (AMA): In which all lag terms are given the same weight of $1/n$.

Exponentially-weighted moving average (EWMA): In which the weight assigned to each lag term exponentially decreases with age, encoding the intuition that more recent samples are more relevant than older ones.

Gaussian-weighted moving average (GWMA): In which lag weights are assigned according to a Gaussian distribution. Values closer to the mean are given larger weights than those farther away, increasing robustness to outliers.

5.1.2. Autoregressive Integrated Moving Average (ARIMA)

ARIMA is a generalization of the Autoregressive Moving Average (ARMA) model. The intuition behind ARMA is that not only can recent variable values be used to predict future ones, but also that past forecasting errors can help correct previously-observed prediction biases.

ARMA forecasts a variable at time t, y_t, as the sum of two polynomials. The autoregressive AR(p), which is a linear combination of the last p variable observations, and the moving average MA(q), which is a linear combination of the last q prediction errors. An ARMA(p, q) model can be written as:

$$y_t = \alpha + \sum_{i=1}^{p} \beta_i \, y_{t-i} + \sum_{i=1}^{q} \phi_i \, \epsilon_{t-i} + \epsilon_t \qquad (12)$$

where α is a constant, β_i is observation y_{t-i}'s weight, ϕ_i is residual error ϵ_{t-i}'s weight, and ϵ_t is white noise.

ARMA assumes a stationary series, i.e., one without trends (long term increases or decreases in values) or seasonality (cycles of fixed and known frequency). An autocorrelation analysis of the throughput series from §3 revealed them not to be stationary [27], despite the repetitive mobility patterns used.

ARIMA is a generalized ARMA model that supports a non-stationary mean (though not variance). Its key insight is that even when there are trends, the difference between consecutive values tends to be uncorrelated. With this in mind ARIMA replaces the y_t values in ARMA's formula with differences between values. The simplest version replaces y_t with $y'_t = y_t - y_{t-1}$, the first-order difference. If this is not enough to make the series stationary, the concept can be applied recursively, e.g., the second-order difference would be the difference between the last two differences. The order of differencing, d, is a parameter, yielding an ARIMA(p, q, d) model. Our throughput data only required $d = 1$ to become stationary [27].

ARIMA uses nothing beyond the previous values of the target variable in its forecasts. However, often there are other correlated variables available that could be used to improve the forecast, e.g., distance to AP and signal strength. ARIMAX is an extension that incorporates a linear function of any such exogenous variables of interest into the forecast. It can be expressed as:

$$y_t = \sum_{i=1}^{m} \theta_i \, x_t^{(i)} + \alpha + \sum_{i=1}^{p} \beta_i \, y'_{t-i} + \sum_{i=1}^{q} \phi_i \, \epsilon_{t-i} + \epsilon_t \quad (13)$$

where $x_t^{(1)}, \ldots, x_t^{(m)}$ are the exogenous variables of interest, and $\theta_1, \ldots, \theta_m$ their respective linear coefficients.

The model's parameters can be discovered following the Box-Jenkins method [28].

5.1.3. *Vector Autoregression (VAR)*

ARIMAX uses the exogenous variables' values at time t to predict the value of the target variable at all future times $t+i$. This is a significant limitation, as the correlation between the two can decrease quickly with time. For example, although current distance to the AP is highly correlated with current throughput, it is not correlated with throughput tens of seconds down the line, as the vehicle could have moved significantly in the meantime.

VAR addresses this by incorporating the predictor variables into the model, i.e., making them endogenous. It defines a system of equations, one for each variable of interest, and all of them are forecast together. Each variable is forecast as a linear combination of prior values of all other variables. This lets throughput at time $t+30$ depend on the distance forecast for $t+29$, for example.

A VAR(p) model is characterized by the number of lag terms considered. An n-dimension VAR(1) model can be written as:

$$\begin{aligned} y_{1,t} &= c_1 + \phi_{11,1}\, y_{1,t-1} + \ldots + \phi_{1n,1}\, y_{n,t-1} + \epsilon_{1,t} \\ &\ldots \\ y_{n,t} &= c_n + \phi_{n1,1}\, y_{1,t-1} + \ldots + \phi_{nn,1}\, y_{n,t-1} + \epsilon_{n,t} \end{aligned} \quad (14)$$

where $c_1, \ldots, c_n$ are constants, $\phi_{ij,1}$ is the weight of $y_{j,t-1}$ on $y_{i,t}$, and $\epsilon_{1,t}, \ldots, \epsilon_{n,t}$ depict white noise.

In our analysis we ran a 3-dimension VAR, featuring throughput, distance to the AP, and signal strength.

5.2. *Mobility clustering-based forecasting*

The channel's signal-to-noise ratio determines the achievable data rate and hence, in a single-user environment, throughput. The vehicle's position defines distance to the AP and the presence and nature of obstacles, which together determine signal attenuation. Thus it makes sense to use mobility, which determines future position, to predict throughput.

With this in mind Meireles et al. [6] presented a scheme that clusters historical network performance data, such as throughput, by a combination of mobility-related variables, and then averages it to make a forecast. The clustering can be done for any combination of variables, but the following two were shown to perform well:

MRDP: Matches the road identifier (R), direction of movement (D), and position relative to the access point (P).

MRDP+SQ: Combines MRDP with signal quality, as defined by the Received Signal Strength Indicator (RSSI), which can be seen as an indirect measure of mobility.

Clusters are divided into a number of buckets equal to the number of seconds into the future we want to forecast. For example, for a forecasting window win_f of 10 s a cluster c_x would have ten buckets $c_x b_{t+1}$ through $c_x b_{t+10}$. Each bucket is meant to hold throughput samples observed i seconds after the vehicle's mobility matched the cluster it belongs to, as per Fig. 6. The intuition is that vehicles with the same present mobility will tend to be at the same location a few seconds from now. And we already covered how location is a good heuristic for throughput.

When a new throughput estimate is received, it is added to the corresponding buckets of the clusters matching the client's mobility over the last win_f seconds. For example, if the vehicle's mobility matched cluster c_x 2 s ago, and cluster c_y 1 s ago, the new sample would be added to buckets $c_x b_{t+2}$ and $c_y b_{t+1}$.

To forecast throughput for time $t+i$, first the current mobility is used to find the correct cluster c_x. Then, the forecast is computed as the average of all samples in bucket $c_x b_{t+i}$. Meireles et al. used an arithmetic average, but others can be used. In fact, we considered two types of Gaussian-weighted averages as alternatives:

1. One that assigns larger weights to samples closer to the observed throughput's mean, hence reducing the impact of outliers.

2. One that assigns larger weights to samples collected closer to the geographical center of the bucket — to leverage this knowledge in the computation phase.

5.3. *Evaluation*

5.3.1. *Setup and model parameterization*

The models' performance was evaluated through the Mean Absolute Scaled Error (MASE) metric [29]. Mean Absolute Error (MAE) is the delta between the forecast and true values. MASE is the ratio between two MAEs. That of the model and that of an in-sample naïve random walk model, where we assume throughput remains constant through time. Thus, an MASE of less than one

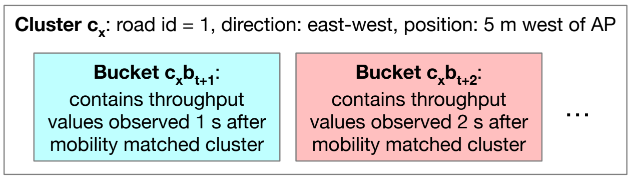

Cluster c_x: road id = 1, direction: east-west, position: 5 m west of AP
Bucket $c_x b_{t+1}$: contains throughput values observed 1 s after mobility matched cluster
Bucket $c_x b_{t+2}$: contains throughput values observed 2 s after mobility matched cluster
...

Figure 6: Internal structure of an example MRDP cluster.

indicates better than naïve performance. Furthermore, the fact that it is scaled lets us more easily compare performance between 802.11n, ac, and ad, which have significantly different absolute throughputs.

Time-series forecasting requires periodic sampling, and mobility-based clustering requires that both training and test data cover the same geographical area. Therefore, our evaluation was performed using only one of our datasets, the Porto dataset from §3.1. The data was sorted by increasing timestamp and then split into two halves. The first was used for training the models that require it (ARIMA(X), VAR, and mobility-based clustering), while the second was used for testing.

We now describe parameterization for the different types of models under evaluation.

The basic moving average models require no training. We tested various different averages, each of them with between 5 and 30 lag terms. The Exponentially-weighted (EWMA) and Gaussian-weighted (GWMA) moving averages with 5 and 10 lag terms performed similarly and better than the others. We opted for the 5-term models due to their lower complexity.

Parameters for the ARIMA(X) models were calculated by following the Box-Jenkins [28] method. The first step is to check for stationarity and seasonality. Seasonality was not observed, obviating the need for a seasonal ARIMA variant. All three time series (802.11n/ac/ad) were found to be non-stationary, but became so with the application of first-order differencing ($d = 1$). Then, the Akaike (AIC) and Bayesian (BIC) Information Criteria were used to select the best p (number of lag terms) and q (number of error terms) parameters, summarized in Tab. 3. Finally, the model weights were determined through Maximum Likelihood Estimation (MLE).

Time series	ARIMA(p,q,d)	ARIMAX(p,q,d)
802.11n	4,1,1	4,1,1
802.11ac	2,1,1	2,1,1
802.11ad	2,1,1	5,1,1

Table 3: Optimal p (number of lag terms), q (number of error terms), d (order of differencing) for the ARIMA(X) models.

The exogenous variables used by ARIMAX were distance to AP and RSSI. For VAR the same two variables were considered endogenous, along with throughput. p values between 1 and 12 were evaluated for VAR. The best performing models were VAR(10) for 802.11n and 802.11ac, and VAR(7) for 802.11ad.

Parameterization of the mobility-based clustering models revolves around selecting the clustering criteria and the average function for forecasting. We experimented with various sets of clustering criteria, but MRDP and MRDP+SQ performed the best. The different clustering components were specified as follows:

Road id: Not considered because our Porto dataset only features one road.

Direction: The vehicle's heading was discretized into four segments of 90° each, corresponding t o the four cardinal directions.

Position relative to AP: Was discretized with a 10 m resolution.

RSSI: Was discretized with a 10 dB m resolution.

In terms of average functions used for forecasting, we evaluated arithmetic average, Gaussian-weighted averages based on throughput and geographical coordinates, and finally a non-parametric distribution estimated through Kernel Distribution Estimation (KDE) with a Gaussian kernel. The simple arithmetic average performed best for 802.11n and ac, with the throughput-based Gaussian average surpassing the other options for 802.11ad.

Additional details on our setup and model parameterization can be found in the related master's thesis [27].

5.3.2. Results

We now present and compare the results obtained by the best-performing variant of each forecasting model.

We evaluated three different scenarios:

Scenario A: Real, measured throughput values were used to both train and test the models. The naïve baseline for the MASE computation consisted of real throughput values also. This scenario let us evaluate forecasting performance in isolation.

Scenario B: SR-UKF-estimated throughput values were used to both train and test the models. The naïve baseline for the MASE computation consisted of real throughput values. This let us evaluate how estimation and forecasting work together.

Scenario C: Same as scenario B, but with a more realistic MASE baseline consisting of SR-UKF-estimated throughput values.

Fig. 7 shows the mean MASE results as a function of the amount of time into the future the forecast pertains to, i.e., the time horizon, for all scenario and network (802.11n, ac, and ad) combinations.

Let us focus first on scenario A, which isolates the forecasting component. Interestingly, the naïve baseline, which predicts throughput will remain constant, outperformed basically all other models for $t + 1$. However, in general, the mobility-based clustering algorithms performed the best across all three networks, getting close to 0.5 MASE for some time horizon, network combinations.

VAR was second best, being quite competitive for 802.11ac, but underperforming the clustering algorithms for the n and ad networks. ARIMA was next in the ranking, but it only outperformed the baseline for 802.11n, with the same being true for the basic moving average algorithms. Despite taking distance and signal strength

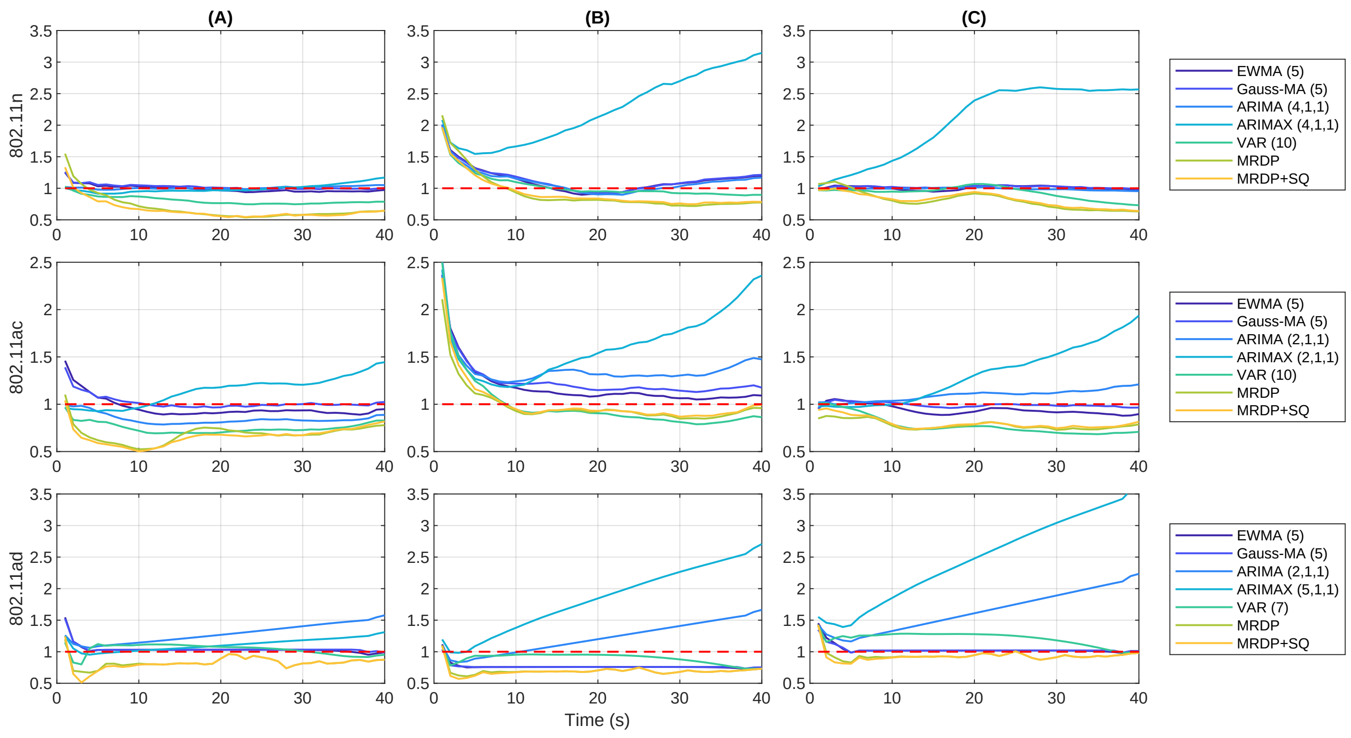


Figure 7: Mean Absolute Scaled Error (MASE) forecasting performance results - (A) calculated using actual throughput values as lags; (B) calculated using SR-UKF-estimated throughput values but scaled based on the actual throughput; (C) calculated and scaled using SR-UKF-estimated throughput values.

into account, ARIMAX underperformed ARIMA and the baseline across all networks.

Consider now scenario B, where forecasts are made based on throughput estimated by our SR-UKF algorithm, but compared against a measured throughput baseline. Naturally, the estimation error propagates into the forecasting phase, increasing MASE across the board. The different algorithms' relative performance remained broadly similar, with the MRPD algorithms performing the best, followed by VAR. They surpassed the constant measured throughput baseline from $t+1$ onwards for 802.11ad, and $t+9$ onwards for 802.11n and ac.

ARIMA and specially ARIMAX performed poorly. MASE for the latter increased somewhat linearly with time horizon, ending up more than 2.5 times that of the baseline after 40 s, for all networks. This indicates that the use of distance and signal strength as exogenous variables was counterproductive. The reason is that, even when forecasting throughput for time $t+40$, ARIMAX uses exogenous variable measurements from time t, long after they have lost their relevance. The approach taken by VAR, which forecasts the evolution of distance and signal strength and then uses them to forecast throughput proved much more effective.

Scenario C represents a fairer challenge, as the baseline is now estimated throughput, not measured. The biggest change from this pertains to short time horizons for 802.11n and ac, with the forecasting algorithms being competitive with the baseline from $t+1$ onwards, instead of $t+9$. This eliminates the rationale for a potential hybrid approach that would use currently-estimated throughput as the forecast for short time horizons. Comparing scenarios B and C, that would only be beneficial if measured values were used, which are in practice unavailable.

5.4. Discussion

Let us now take a step back and consider the problem holistically. In our experiments, the vehicle remained connected to the 802.11n and ac networks throughout. Thus modelling throughput as a time series makes sense. In the real world however, connectivity will be sporadic. Long disconnection periods will be interspersed by short high-throughput bursts, as the vehicle moves in and out of range. More akin to what happened with the shorter range 802.11ad network in our experiments. This makes it hard for traditional time series algorithms to find patterns.

The use of additional correlated variables, such as distance and signal strength, can help, as demonstrated by VAR's performance. But the best overall results came from the mobility clustering algorithms, which do not rely on the existence of a time series. The results of MRDP and MRDP+SQ were almost indistinguishable. This makes the former preferable, as simpler models require less data to train and are less prone to overfitting.

6. Conclusions and future work

In this paper we explored the problems of estimating current throughput and forecasting its future evolution, to support network selection in vehicular Wi-Fi access.

We combined Symbolic Regression (SR) with an Unscented Kalman Filter (UKF) to create a lightweight throughput estimation model that uses nothing but passively-observable variables. Benchmarking on our experimental dataset against common ML models, including more complex ones, showed the UKF-SR model to generally outperform them.

For forecasting future throughput, we explored traditional time-series methods such as ARIMA and VAR, along with the mobility-based clustering MRDP family of algorithms. The latter is domain-specific, leveraging the fact that throughput is tightly coupled with the vehicle's location. Consequently, it outperformed the remaining options, regardless of whether measured or estimated throughput values were used.

Our MRDP forecasting algorithms cluster data based on mobility features and signal quality. This is sufficient for single-user scenarios such as the one tested. However, as shown by our UKF-SR equations (Eqs. (1), (2)), in multi-user environments throughput is significantly influenced by channel load. The forecasting algorithm is flexible in that the clustering can be performed using any set of variables deemed relevant. So it is both possible and logical to add a network load indicator to that set, which we hope to do in the future.

At first we planned to use the number of active stations for this purpose. However it is hard to obtain passively:

1. Counting stations requires listening to all frames. This necessitates putting the device into promiscuous mode, which precludes regular communication.
2. The number could be underestimated due to the presence of hidden terminals.

One possibility is to use a more indirect load indicator. For example, time-of-day or day-of-the-week could capture seasonality patterns in load and are passively observable. However, for best results we would like to employ a more direct metric. Fortunately, the Quality-of-Service amendment to IEEE Wi-Fi, 802.11e [30], specifies how APs can include a channel utilization percentage in their periodic beacons. We would like to incorporate that value into our throughput estimation and forecasting.

Another avenue for future work is the following. The mobility clustering-based protocols couple the forecasting of the vehicle's future position with throughput forecasting. We plan to split these into two separate steps. This will let us pick different algorithms for each of them, potentially improving performance.

Finally, we plan on combining the best-performing estimation and forecasting methods with the network selection strategies first introduced by Meireles et al. [6], to create an actual prototype system capable of intelligently choosing between networks in real-time.

References

[1] M. Harris, The radical scope of Tesla's data hoard, IEEE Spectrum 59 (2022) 40–45.
URL https://spectrum.ieee.org/tesla-autopilot-data-scope

[2] Cisco Annual Internet Report (2018-2023), (accessed on 2022-12-11) (2020).
URL https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] A. Rodrigues, P. Steenkiste, A. Aguiar, Wi-Fi Assist: Enhancing Vehicular Wi-Fi Connectivity with an Infrastructure-driven Approach (2022). arXiv:2207.04547.

[4] Agile Content Group, Fon Wi-Fi, (accessed on 2023-11-11) (2023).
URL https://fon.com

[5] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, R. K. Sinha, Can Accurate Predictions Improve Video Streaming in Cellular Networks?, in: 16th International Workshop on Mobile Computing Systems and Applications (HotMobile), ACM, 2015, pp. 57–62. doi:10.1145/2699343.2699359.

[6] R. Meireles, A. Rodrigues, A. Stanciu, A. Aguiar, P. Steenkiste, Exploring Wi-Fi Network Diversity for Vehicle-To-Infrastructure Communication, in: 2020 IEEE Vehicular Networking Conference (VNC), 2020, pp. 1–8. doi:10.1109/VNC51378.2020.9318407.

[7] Y. Liu, J. Y. B. Lee, An Empirical Study of Throughput Prediction in Mobile Data Networks, in: 2015 IEEE Global Communications Conference (GLOBECOM), 2015, pp. 1–6. doi:10.1109/GLOCOM.2015.7417858.

[8] D. Teixeira, R. Meireles, A. Aguiar, Wi-Fi Throughput Estimation for Vehicle-to-Network Communication in Heterogeneous Wireless Environments, in: 2023 18th Wireless On-Demand Network Systems and Services Conference (WONS), 2023, pp. 24–31. doi:10.23919/WONS57325.2023.10061940.

[9] M. Li, M. Claypool, R. Kinicki, WBest: A bandwidth estimation tool for IEEE 802.11 wireless networks, in: 2008 33rd IEEE Conference on Local Computer Networks (LCN), 2008, pp. 374–381. doi:10.1109/LCN.2008.4664193.

[10] M. Mathis, J. Semke, J. Mahdavi, T. Ott, The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, SIGCOMM Comput. Commun. Rev. 27 (3). doi:10.1145/263932.264023.

[11] A. Samba, Y. Busnel, A. Blanc, P. Dooze, G. Simon, Instantaneous throughput prediction in cellular networks: Which information is needed?, in: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 624–627. doi:10.23919/INM.2017.7987345.

[12] Q. He, C. Dovrolis, M. Ammar, On the Predictability of Large Transfer TCP Throughput, ACM SIGCOMM Computer Communication Review 35 (4) (2005) 145–156. doi:10.1145/1090191.1080110.

[13] B. Wei, W. Kawakami, K. Kanai, J. Katto, S. Wang, TRUST: A TCP Throughput Prediction Method in Mobile Networks, in: 2018 IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1–6. doi:10.1109/GLOCOM.2018.8647390.

[14] Outlines, Car blueprints and vector drawings, (accessed on 2022-06-16) (2022).
URL https://getoutlines.com

[15] TP-Link, AD7200 Multi-Band Wi-Fi Router Datasheet, (accessed on 2022-11-13) (2018).
URL https://static.tp-link.com/2018/201809/20180912/AD7200%202.0%20Datasheet.pdf

[16] D. Steinmetzer, D. Wegemer, M. Hollick, Talon Tools: The Framework for Practical IEEE 802.11ad Research, (accessed on

2022-05-30) (2017).
URL https://seemoo.de/talon-tools

[17] MikroElektronika GNSS 5 click, (accessed on 2022-06-22) (2022).
URL https://www.mikroe.com/gnss-5-click

[18] D. Teixeira, R. Meireles, A. Aguiar, WiPerf Vehicular Wi-Fi Performance Monitoring Dataset (2022). doi:10.5281/zenodo.6761916.
URL https://doi.org/10.5281/zenodo.6761916

[19] O. K. Isik, J. Hong, I. Petrunin, A. Tsourdos, Integrity Analysis for GPS-Based Navigation of UAVs in Urban Environment, Robotics 9 (3). doi:10.3390/robotics9030066.

[20] R. Meireles, A. Rodrigues, A. Stanciu, A. Aguiar, P. Steenkiste, A Dataset for Exploring Wi-Fi Network Diversity in Vehicle-to-Infrastructure Communication (2020). doi:10.5281/zenodo.6884095.
URL https://doi.org/10.5281/zenodo.6884094

[21] D. P. Searson, GPTIPS 2: An Open-Source Software Platform for Symbolic Data Mining, in: A. H. Gandomi, A. H. Alavi, C. Ryan (Eds.), Handbook of Genetic Programming Applications, Springer International Publishing, 2015, pp. 551–573. doi:10.1007/978-3-319-20883-1_22.

[22] G. F. Smits, M. Kotanchek, Pareto-Front Exploitation in Symbolic Regression, in: U.-M. O'Reilly, T. Yu, R. Riolo, B. Worzel (Eds.), Genetic Programming Theory and Practice II, Springer US, 2005, pp. 283–299. doi:10.1007/0-387-23254-0_17.

[23] J. Karedal, N. Czink, A. Paier, F. Tufvesson, A. F. Molisch, Path Loss Modeling for Vehicle-to-Vehicle Communications, IEEE Transactions on Vehicular Technology 60 (1) (2011) 323–328. doi:10.1109/TVT.2010.2094632.

[24] S. J. Julier, J. K. Uhlmann, New extension of the Kalman filter to nonlinear systems, in: I. Kadar (Ed.), Signal Processing, Sensor Fusion, and Target Recognition VI, Vol. 3068, International Society for Optics and Photonics, SPIE, 1997, pp. 182–193. doi:10.1117/12.280797.

[25] B. J. Odelson, M. R. Rajamani, J. B. Rawlings, A new autocovariance least-squares method for estimating noise covariances, Automatica 42 (2) (2006) 303–308. doi:10.1016/j.automatica.2005.09.006.

[26] R. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, 2nd Edition, OTexts, Australia, 2018.

[27] D. Teixeira, Opportunistic Wi-Fi network selection in heterogeneous vehicular wireless networks for detecting VRUs through edge computing, Master's thesis, University of Minho (2022).
URL https://hdl.handle.net/1822/84496

[28] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis: forecasting and control, Wiley, 2015.

[29] R. J. Hyndman, A. B. Koehler, Another look at measures of forecast accuracy, International Journal of Forecasting 22 (4) (2006) 679–688. doi:https://doi.org/10.1016/j.ijforecast.2006.03.001.

[30] IEEE, IEEE Standard for Information technology–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003) (2005) 1–212doi:10.1109/IEEESTD.2005.97890.