# WiChoose: Practical Network Selection for Wi-Fi Vehicle-to-Infrastructure Communication

1st Rui Meireles
*Department of Computer Science*
*Vassar College*
Poughkeepsie, NY, USA
rui.meireles@vasar.edu

2nd Vinícius Abrunhosa
*Instituto de Telecomunicações*
*University of Porto*
Porto, Portugal
abrunhosavinicius@gmail.com

3rd Ana Aguiar
*Instituto de Telecomunicações*
*University of Porto*
Porto, Portugal
anaa@fe.up.pt

***Abstract*—The proliferation of Wi-Fi hotspots enables opportunistic vehicular access. In urban areas there can be multiple networks in range, necessitating a choice. Network heterogeneity and fast vehicular mobility make this difficult. We present WiChoose, which is both a practical network selection system and a framework for the evaluation of selection strategies. It monitors available networks and decides how to switch between them, in order to maximize data transfer. Vehicle location is a key network performance determinant, and current mobility predicts future location. We have thus previously proposed clustering performance data by mobility features, to help forecast future network performance given the vehicle's current mobility. WiChoose refines this strategy by additionally prioritizing recent observations over older ones. As a testing framework, WiChoose is highly configurable and extensible. It can run multiple selection strategies in parallel and measure actual throughput, for comparison purposes. We used WiChoose to experimentally evaluate our proposed scheme in a realistic vehicular setting featuring IEEE 802.11n and ad networks. It successfully alternated between the two networks, transferring 95 % of the total amount of data that a future-seeing oracle scheme could have.**

***Index Terms*—C.2.1.k Wireless communication, C.2.8.c Mobile communication systems**

## I. Introduction

Vehicles have become computers on wheels. They increasingly require network connectivity for navigation, over-the-air updates, telemetry, and more. Vehicle-to-Infrastructure (V2I) communication currently uses mostly cellular connections, which are becoming ever more strained. Thus, Wi-Fi hotspots accessible from the street in urban areas create a good opportunity for data offloading. A large-scale 2016-19 study [2] performed in the city of Porto, Portugal, found 80 % of streets to be covered by one or more public hotspots.

In cities it is common for multiple networks to be available, which poses the question of which one to use at any given time. We focus on this question, specifically for data-intensive but delay-tolerant applications, such as the download of over-the-air software updates, or the upload of vehicle sensor data to the cloud for data mining or machine learning purposes [3].

This work was supported by: i) the European Union/Next Generation EU, through Programa de Recuperação e Resiliência (PRR) [Project Nr. 29: Route 25 (02/C05-i01.01/2022.PC645463824-00000063)]; ii) Fundação para a Ciência e Tecnologia, I.P. (FCT) under Project UIDB/50008/2020 [1], funded by FEDER through COMPETE.

Under such a scenario, clients will want to maximize the total amount of data transferred over Wi-Fi, which equates to maximizing the sum of throughputs over time. The problem of picking the best network can then be divided into:

1) Estimating current throughput for each network.
2) Forecasting throughput evolution over time.
3) Deciding when and to what network to switch.

Estimating throughput is challenging. For a given network, signal strength determines the achievable data rate. However, different Wi-Fi standards yield different data rates for a given signal strength. Also, since the wireless medium is shared, network load will affect how data rate translates to throughput. One option is to measure throughput directly with probe traffic, but that adds unwanted load. Previously [4], we applied Symbolic Regression (SR) to a set of throughput measurements to find estimation formulas based on passively-observable variables, such as signal strength and user count. Testing showed them to be competitive against more sophisticated models, while being computationally inexpensive.

After estimating current throughput, we need to forecast its evolution over time. Intuitively, location is a good throughput predictor and, combined with velocity, a good predictor of future location. Thus we introduced a family of algorithms [5] that forecast future throughput as the mean of previously-observed throughputs given the current set of mobility features, which include the vehicle's position and direction of movement. A trace-based evaluation [4] compared them favourably against traditional domain-agnostic time series algorithms such as Autoregressive Integrated Moving Average (ARIMA) and Vector Autoregression (VAR).

Once throughput has been forecast for all networks, network selection can occur. We devised an algorithm that computes the optimal switching schedule given the forecasts. In a trace-based evaluation [5], it created schedules capable of offloading 90 % of the data of a theoretical oracle algorithm with access to real future throughput values.

We now evolve and implement these components to create a practical network selection system we call WiChoose. Unlike our previous work, which ran offline, WiChoose is capable of picking networks on-the-fly, at runtime. It is written in C++ and runs on commodity Linux-based devices.


ISBN: 978-3-903176-71-3

Additionally, WiChoose is built as a framework for experimental evaluation of network selection schemes. Its object-oriented design is highly parameterizable and extensible, allowing for different selection strategies to be added and tested in parallel. Further, it collects actual throughput measurements for each network, supporting quantitative comparison between schemes.

We leveraged WiChoose to experimentally assess our own network selection strategy. We drove a circuit around a simulated stoplight while choosing between two different Wi-Fi networks, one 802.11n and one 802.11ad. These networks offer different trade-offs between communication range and throughput. WiChoose proved capable of switching between them effectively, offloading 95 % of the upper bound amount set by a theoretical all-knowing oracle scheme.

In summary, we make the following contributions:

- Describe our network selection process and prove it makes optimal choices, given the provided throughput information (§II).
- Present WiChoose, a practical Wi-Fi-based V2I network selection system and testing framework (§III).
- Use WiChoose to experimentally evaluate our network selection strategy on a realistic vehicular scenario (§IV).

## II. Network selection workflow

This section details our network selection process, which consists of estimating current throughput, forecasting its evolution, and then using the forecasts to pick the network that can transfer the most data.

### A. Throughput estimation

Our goal for throughput estimation was to create a computationally inexpensive model that relies only on passively-observable variables. To this end we took an experimental vehicular dataset and applied Symbolic Regression (SR) to derive throughput formulas for different Wi-Fi standards.

The dataset used [6] pertains to experiments where a vehicle drove a circuit around multiple static and colocated Access Points (APs), including 802.11n and 802.11ad ones, while downloading data from them. It consists of a 1 Hz-resolution log where each entry features a throughput measurement, the vehicle's coordinates and speed, a Received Signal Strength Indicator (RSSI), and the number of active clients.

Symbolic regression consists of searching the parameter space for the combination of state variables, operators and functions that best fits the data. To prevent overfitting, fitness was defined as the combination of Mean Absolute Error (MAE) and number of expression terms. The GPTIPS2 genetic-programming algorithm [7] was used, yielding:

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

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

$\hat{tput}_n$ and $\hat{tput}_{ad}$ are the throughput estimates for 802.11n and ad, in Mbps; $RSSI$ is in dBm; $s$ is the vehicle's speed in m/s; and $N_{users}$ the number of active users.

### B. Throughput forecasting

To support network selection our goal is to, at time $t$, forecast throughput for $t+0$, $t+1$, …, $t+win_f-1$, where $win_f$ is the length of the forecasting window of interest[1].

Position is a key throughput indicator. For a given network, signal strength dictates the achievable data rate and hence, in a single-user environment, throughput. The vehicle's position defines distance to the AP and topography-derived line-of-sight conditions, which together determine signal attenuation, and thus throughput. Current mobility predicts future position and therefore, indirectly, also future throughput.

Our forecasting approach, which we have shown to perform well [4], is to cluster historical throughput data by a set of mobility features and then average the samples to compute a forecast. We codify mobility as the combination of the following features: road identifier (R), direction of movement (D), position relative to the AP (P), and binary speed indicator (S), i.e., *low/stopped* or *high/moving*. The latter is useful for unstable networks that are only usable at low speeds, e.g., 802.11ad. We call this an MRDPS clustering, after its different components.

Since WiChoose currently targets a single-user scenario, these features suffice. However, the model can easily be expanded to account for the effect of channel sharing among multiple users. For example, by adding channel busy ratio, or user count, to the predictor feature set.

Each MRDPS cluster is split into a number of buckets equal to the forecasting window length. For example, given a 10 s window, a cluster $c_x$ would contain ten buckets $c_x b_{t+0}$ through $c_x b_{t+9}$. Bucket $c_x b_{t+i}$ would hold throughput samples observed $i$ seconds after the vehicle's mobility matched cluster $c_x$, as depicted in Fig. 1.

At each time $t$, a new estimate is made and added to the corresponding buckets of the clusters matching the client's mobility over the last $win_f$ seconds. Thus, if the vehicle's mobility matched cluster $c_x$ at $t-2$ and cluster $c_y$ at $t-1$, the new value would be added to buckets $c_x b_{t+2}$ and $c_y b_{t+1}$.

To the previous clusters that summarize the entire system's history, we add a special "current cluster", $c_{cur}$. It holds

[1]The window starts at $t+0$ to also support measurement-based decisions. While an estimate for time $t$ can be obtained at time $t$, a measurement can only be obtained at time $t+1$. Hence the need to forecast for $t+0$.

**Cluster $c_x$**: road id: 1, position: 10m east of AP, heading: west, speed: low

| Bucket | Contents |
|---|---|
| $b_{t+0}$ | throughput values seen 0 s after cluster match |
| $b_{t+1}$ | throughput values seen 1 s after cluster match |
| … | |
| $b_{t+9}$ | throughput values seen 9 s after cluster match |

Fig. 1: Example MRDPS cluster internal structure.


ISBN: 978-3-903176-71-3

the throughput samples collected since the vehicle's mobility started matching the cluster it presently does. For example, if the matched cluster was $c_x$ at time $t-2$, and then $c_y$ at $t-1$ and $t$, $c_{cur}$ will contain samples from these last two timestamps, and nothing else. As as consequence of its definition, $c_{cur}$ is reset every time the matched cluster changes. This cluster is a refinement relative to prior work that lets us privilege newer data over older data in the forecasting process, e.g., to capture a transient line-of-sight obstruction.

The throughput forecast for time $t+i$ is determined by:

1) Finding the right bucket $b$. If the special current cluster $c_{cur}$ contains samples for offset $i$, then $b = c_{cur}b_{t+i}$. Otherwise, current mobility information is used to find the matching regular cluster, $c_x$, and $b = c_x b_{t+i}$.
2) Computing the forecast as the average of all samples in bucket $b$. Currently WiChoose supports arithmetic mean and simple exponential moving average (i.e., one history term), but any other summarization function could be employed. If no data exists, zero is forecast.

### C. Network selection decision

The aim is to select networks in a way that maximizes the total amount of data offloadable over Wi-Fi. To this end, each passing second we use the current throughput estimates and forecasts over the forecasting window $win_f$ to estimate the amount of data offloadable over that period for each network. Ultimately, the network with the largest estimate is chosen.

Let $t_0$ be the current time. The maximum offloadable data from time $t_i$ on ($i \geq 0$), given a current network $cnet$, is:

$$Data_{cnet}[t_i] = \begin{cases} Tput_{cnet}[t_i] \\ + \\ \max\limits_{\forall onet} fd(cnet, onet, t_i) & \text{if } i < win_f \\ \\ 0 & \text{if } i \geq win_f \end{cases} . \quad (3)$$

$Data_{cnet}[t_i]$ is the sum of the estimated current throughput for $cnet$, $Tput_{cnet}[t_i]$, with the maximum amount of data that can be offloaded in the future. The latter is given by maximizing function $fd(cnet, onet, t_i)$, which represents the future offloadable data if a switch from current network $cnet$ to network $onet$ is started at time $t_i$, over the available networks. It is defined as:

$$fd(cnet, onet, t_i) = \begin{cases} Data_{onet}[t_{i+1+ot}] & \text{if } onet \neq cnet \\ Data_{cnet}[t_{i+1}] & \text{if } onet = cnet \end{cases} . \quad (4)$$

The first case represents an actual network switch, which implies an outage period of $ot$ seconds, hence why the same amount of time is skipped in the computation. The second case corresponds to staying on the current network, in which case data transfer continues uninterrupted.

$Data_{cnet}[t_i]$'s recursive definition lends itself to an efficient dynamic programming-based implementation that starts by computing $Data_{cnet}[t_{win_f-1}]$ and then works backwards towards $Data_{cnet}[t_0]$, saving the intermediate values for later use in the process. This is exactly what WiChoose does, as depicted by the pseudocode in Alg. 1.

The procedure takes in the current network $cnet$, the forecasting window length $win_f$, the number of networks $nnets$ (each network is uniquely identified by a number in $[0, nnets-1]$), and a $win_f \times nnets$ matrix of throughput values $tputFore$. $tputFore[t_i][net]$ holds the estimate/forecast for network $net$ at time $t_i$ (assume indices range from $t_0$ to $t_{win_f-1}$ for the matrix's first dimension, and from 0 to $nnets-1$ for the second).

The code loops down from $t_{win_f-1}$ to $t_0$. For each time $t_i$ it goes through each network $net$, computes $Data_{net}[t_i]$ as per Eq. 3, and saves the result in a dynamic programming memory, $data$. This means that, when processing $t_i$, the values for $t_{i+1}$, $t_{i+2}$, etc, are all available. As it goes through each $(t_i, net)$ pair, it also records which network should be used next in order to maximize the amount of offloadable data, $nextNet$.

Once $Data_{cnet}[t_0]$ has been computed, which occurs when $t_i = t_0$ and $net = 0$, processing stops and the next network for that combination is returned. Note that termination is guaranteed because the $\{t_i = t_0,\ net = 0\}$ combination is necessarily generated by the loops of lines 6 and 7.

Moreover, we can prove $Data_{net}[t_i]$, $\forall net, t_i$, is correctly computed, given the input data, by induction on time $t_i$:

```
1  inputs: int cnet, int win_f, int ot, int nnets, int[][] tputFore
2  output: int
3  begin
4    data ← (win_f+ot+2) × nnets matrix of zeros          # offloadable data memory for each (time, network) pair
5    for t_i from t_{win_f-1} downto t_0:                 # for each time in forecasting window (reverse order)
6      for net from 0 upto nnets-1:                       # find offloadable data for each (time, network) pair
7        maxFdata ← data[t_{i+1}][net]                    # future offloadable data if we stay in network
8        nextNet ← net                                    # stores network yielding maxFdata
9        for onet from 0 upto nnets-1:                    # check other networks
10         fdataOnet ← data[t_{i+1+ot}][onet]             # future offloadable data if we initiate switch to onet
11         if fdataOnet > maxFdata:                       # update future data if switching to onet is better
12           maxFdata ← fdataOnet
13           nextNet ← onet                               # next network is now onet
14       data[t_i][net] ← tputFore[t_i][net] + maxFdata   # save offloadable data value
15       if t_i == t_0 and net == cnet:                   # is this the (time, network) pair of interest?
16         return nextNet
17 end
```

Alg. 1: Throughput forecasts-based network selection algorithm.


ISBN: 978-3-903176-71-3

**Base case** ($t_i = t_{win_f-1}$)**:** Since the time horizon is limited to $t_0 + win_f - 1$, all choices yield the same amount of data, $Tput_{net}[t_{win_f-1}]$. Our algorithm correctly computes this. Since the $data$ matrix is zero-filled, lines 7 and 10 will set $maxFdata$ to zero. Line 14 will then set $Data_{net}[t_{win_f-1}] = Tput_{net}[t_{win_f-1}] + 0$ as desired.

**Inductive case** ($t_i \to t_{i-1}$)**:** Assume $\forall net$, $Data_{net}[t_i]$ has been computed correctly and we are now computing $Data_{anet}[t_{i-1}]$ for a network $anet$. $Data_{anet}[t_{i-1}]$ is set in line 14 to be $Tput_{anet}[t_{i-1}] + maxFdata$, which agrees with Eq. 3, as long as $maxFdata$ is equal to $\max_{\forall onet} fd(anet, onet, t_{i-1})$. $maxFdata$ is initialized to $Data_{anet}[t_i]$ in line 7. We now distinguish two cases:

1) Staying on the current network is optimal, meaning $Data_{anet}[t_i] \geq Data_{onet}[t_{i+ot}]$, for $\forall onet \neq anet$. By our inductive hypothesis, all of these data estimates will be correct, therefore the condition of line 11 will always be false, and $maxFdata$ will remain equal to $Data_{anet}[t_i]$, which is correct.
2) Switching networks is optimal, i.e., $\exists dnet \neq anet$ such that $dnet = \arg\max_{onet} fd(anet, onet, t_{i-1})$. Since the loop of line 9 goes through all networks, it will go through $dnet$. By our inductive hypothesis, all data estimates for times $\geq t_i$ will be correct and thus line 11's conditional will ensure that $maxFdata$ is set to $Data_{dnet}[t_{i+ot}]$, which is the desired value.

calculated, and consequently so will $Data_{anet}[t_{i-1}]$.

The correct computation of $Data_{net}[t_i], \forall net, t_i$, implies the same for the specific case of $Data_{cnet}[t_0]$. The next network returned is thus optimal, i.e., the one yielding the maximum future offloadable data.

Running time is $O(win_f \times nnets^2)$, as dictated by the nested loops in lines 5-6. Spatial complexity is determined by the size of the dynamic programming memory $data$ in line 4, which is $\Theta((win_f + ot) \times nnets)$. Since the expected number of networks is small, this is a practical solution.

## III. WiChoose Design and Implementation

This section details WiChoose's structure and features.

### A. Design

WiChoose's main requirements were: i) be able to choose networks at runtime using the logic from §II, and ii) act as a framework to support the evaluation of different selection schemes. In particular, we wanted to compare our scheme against one that used throughput measurements instead of estimates, and against the theoretical optimum. This necessitated the ability to measure actual throughput, and run multiple selection schemes in parallel.

These goals, along with universal ones such as simplicity and modularity, led us to an object-oriented design. Its architecture is depicted in Fig. 2. The system has two nodes:

**Mobile client:** Responsible for choosing the best network according to the available performance information, and for sending data to the access point.

**Access point:** Responsible for receiving data and measuring throughput. Further, it sends the latter as feedback to the client, enabling measurement-based schemes.

Each node runs multiple processes, which communicate through shared memory. We now describe each of them.

*1) Global Navigation Satellite System (GNSS) daemon:* The GNSS daemon is responsible for providing both nodes with mobility and timing information. It does this by parsing the raw National Marine Electronics Association (NMEA) 0183 sentences generated by a GNSS receiver, and writing the extracted information to shared memory. The observer pattern is used to notify interested parties of changes.

*2) Data receiver:* Runs on the access point. It executes one data-receiving thread per available network interface. The number of bytes received over each of them during each second is logged and periodically sent to the mobile client as feedback. Both feedback message frequency and length of throughput history included in each are configurable.

*3) Network selector:* This mobile-client process gathers throughput data and uses it to pick networks. It comprises:

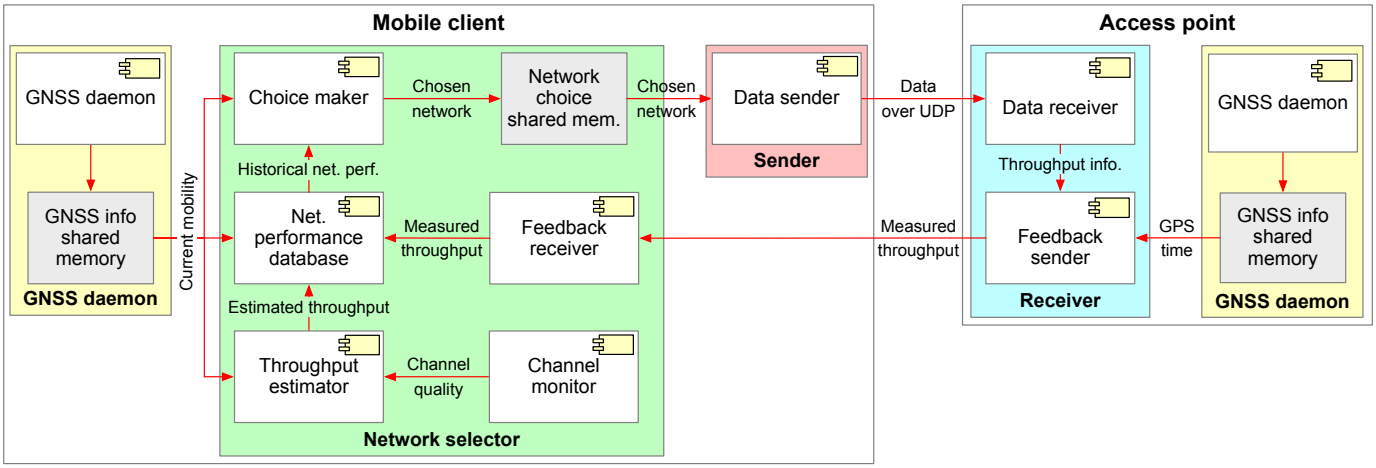



Fig. 2: WiChoose's combined logical and physical system architecture.


ISBN: 978-3-903176-71-3

*a) Network performance database::* Stores throughput measurements and estimates for forecasting. Data is clustered by mobility features, as per the MRDPS algorithm of §II-B.

For performance and simplicity, we created a hash table-based in-memory database. The keys are mobility features and the values are clusters of throughput data, as per Fig. 1.

Data persistence is achieved by writing the database to a file prior to process termination, and reading it back upon restart. The file is human-readable, to facilitate debugging.

All database operations are specified in an abstract interface, allowing for a different implementation, e.g., a relational database, to be easily swapped in if desired.

*b) Throughput estimator::* A thread that, once a second, estimates throughput for each network according to the formulas in §II-A, and saves the results to the database.

Estimation requires three inputs:

**$N_{users}$:** Set to 1 as the prototype features a single client node.
**$s$ (speed):** Taken directly from the GNSS shared memory.
**RSSI:** For simplicity, WiChoose assumes a 1:1 relationship between networks and network interfaces. Therefore we read from file `/proc/net/wireless`, which, on Linux systems, reports RSSI for each interface.

*c) Feedback receiver::* A thread that listens for throughput feedback messages from the data receiver node, extracts any new information, and forwards it to the database.

*d) Choice maker::* A thread that, once a second, applies the algorithm from §II-C to find the network that maximizes the amount of offloadable data within the lookahead period. The algorithm is run twice: one informed by forecasts based on throughput estimates, and another based on measurements. Both are retrieved from the database. Results are written to shared memory for the data sender process to read.

Both strategies are implemented as subclasses of a common superclass, with variance coming from their different implementations of a set of polymorphic methods. More decision strategies can be added by writing new subclasses.

*4) Data sender:* Streams data to the receiver over UDP, to prevent the influence of TCP flow and congestion controls. Since we want to measure throughput, it sends at the fastest possible rate. And as content doesn't matter, the data source is an uninitialized buffer. The process can operate in one of two modes, depending on configuration:

**Stream over all:** Sends data over all available interfaces, using a separate thread for each. Meant for evaluation.
**Stream over best:** Sends data over the best interface, as per shared memory. Meant for a "production" setting.

### B. Implementation

Due to their low cost, we wanted WiChoose to be able to run on commodity Linux-based embedded systems. This led us to choose C++17 as the implementation language, given its low resource use and wide availability.

We ran WiChoose on TP-Link Talon AD7200 [8] routers. These arm-based devices feature multiple Wi-Fi radios in a compact package, simplifying logistics. Since the factory Operating System (OS) is not user-programmable, we replaced it with LEDE-AD7200 [9], a Talon-customized version of the popular OpenWrt GNU/Linux distribution [10].

LEDE-AD7200 was thus the OS we targeted. However, there should be no issues compiling and running WiChoose on other Linux-based systems. For maximum compatibility, besides the C++ standard library, the code only depends on the ubiquitous pthread (libpthread) and real-time (librt) libraries. For multi-threading and shared-memory operations, respectively. Additionally, our makefiles support not just arm LEDE-AD7200 targets, but also native-architecture ones.

For ease of use, all runtime parameters are read from a single configuration file: `/etc/wichoose.conf`.

WiChoose's source is open and available on GitLab [11].

## IV. Experimental Evaluation

This sections describes our evaluation's setup and results.

### A. Setup

Two main goals motivated our experimental setup: (i) exercise the system's ability to hop between networks to maximize data transfer, and (ii) logistical simplicity.

We settled on two networks: one 802.11n, and one ad. The former offers long range and stable but low throughput. The latter the exact opposite. The experiments were done at an industrial park near Porto, Portugal. The AP for both networks was placed next to a crosswalk and the client vehicle drove a circuit around it, as per Fig. 3. Half the times the client approached from the northwest, it stopped at the crosswalk for ∼40 s, to simulate a red stoplight. The other half it drove

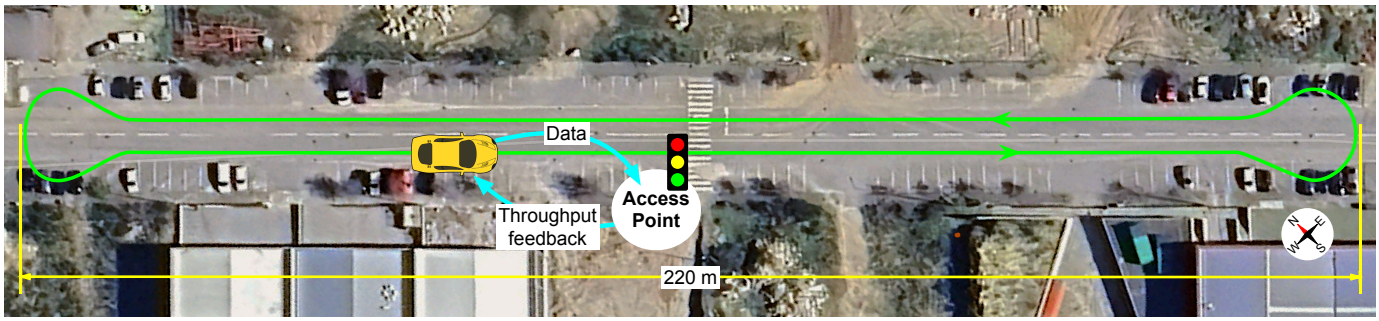



Fig. 3: Physical experiment setup. Access point coordinates: 41.31562042, -8.291383743. Imagery © 2024 Airbus.


ISBN: 978-3-903176-71-3

on through, simulating a green light. We drove a total of 20 laps, in around 30 min. Speed ranged from 0 to 50 km/h.

Talon AD7200 multi-radio routers we used for both AP and client. The AP was placed on a tripod. The client, on the roof of a passenger car. GNSS receivers were equipped, for localization and time synchronization. The forecasting database started out empty. Finally, Tab. I summarizes the most relevant WiChoose configuration parameters.

TABLE I: System configuration used in the experiments.

| Category | Parameter | Value |
|---|---|---|
| Throughput forecasting | Position resolution (m) | 10 |
| | Direction resolution (°) | 180 |
| | Low/high speed threshold (km/h) | 20 |
| | Forecast window length $win_f$ (s) | 40 |
| | Exp. moving avg. smoothing factor $\alpha$ | 0.3 |
| Network selection | Network switch outage time $ot$ (s) | 1 |
| Throughput feedback sender | Sending frequency (Hz) | 2 |
| | Measurement window length (s) | 10 |

In addition to the estimate-based scheme from §II, we ran a variation that uses throughput measurements for forecasting, as a comparison point. Moreover, those throughput measurements were logged so we could also compare against:

**Optimal:** An oracle scheme that uses accurate and complete information to find the perfect network switching schedule. Represents a performance upper bound.

**11n-only:** A scheme that only uses the n network, due to its stability. Represents a performance lower bound.

### B. Results

Tab. II lists the amount of data offloadable by each scheme.

There is much to gain by switching networks, as the 85 % increase from 11n-only to optimal attests. The estimate scheme was very effective, yielding 95 % of the optimal. Interestingly, despite having access to real measured throughput values and not estimates, the measurement variant performed slightly worse, at 93 %.

TABLE II: Total offloadable data comparison.

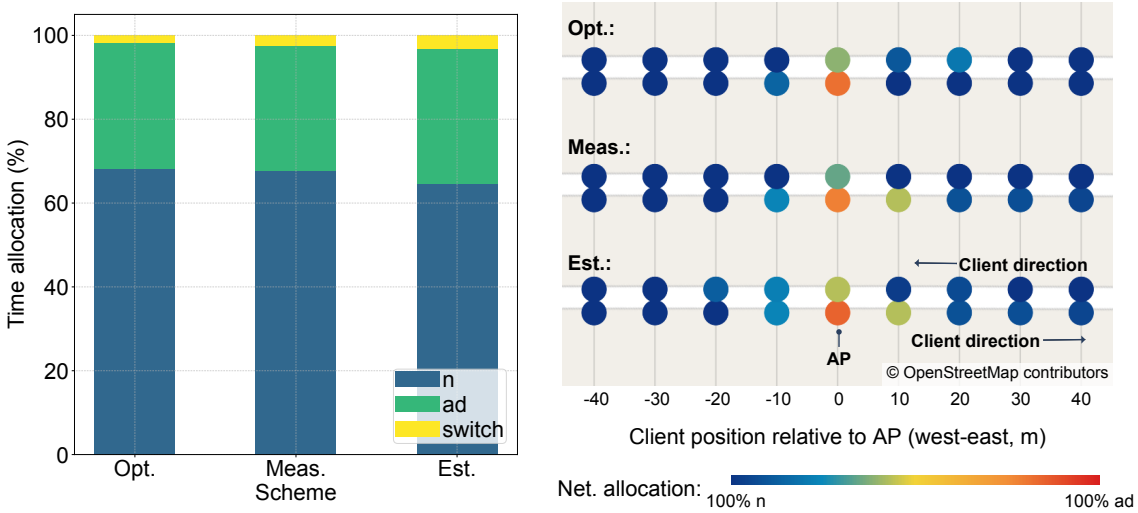
| Scheme | Optimal | 11n-only | Meas.-based | Est.-based |
|---|---|---|---|---|
| Data (GB) | 27.88 | 15.09 | 26.01 | 26.46 |
| Data (% of opt.) | 100 | 54.12 | 93.29 | 94.91 |

To understand these numbers, let us inspect the schemes' behavioral differences. Fig. 4a shows the network allocation distribution, in aggregate, while Fig. 4b does the same by client location. The optimal scheme only uses the ad network when very close to the AP. The estimate scheme slightly overuses the ad network, which accounts for its performance gap relative to the optimal. The measurements scheme, on the other hand, matches the optimal aggregate allocation quite well. Thus, its performance deficit must be timing related.

Fig. 5 provides further evidence. It depicts a timeline slice captured as the client approached the AP on a red light and ad connectivity improved. The optimal scheme switched to ad once and stayed there. The measurement scheme also, but a little too late. The estimate scheme switched to ad twice.

Consider now the root causes of these differences. First, timing. Since the optimal scheme knows the future, it can time switches perfectly. The estimate scheme can not. However, it can quickly react to RSSI changes. In contrast, the measurement scheme suffers from inherent extra delay: throughput measurement for time $t$ can only end upon the transition to time $t + 1$, and then it has to be sent over to the client.

The cause for the estimate-based scheme's slight ad overuse was determined to be estimation error. The mean throughput increase offered by ad was measured as 34 Mbit/s, but estimated to be 61 Mbit/s, or 80 % more. To help pinpoint the cause, Fig. 6 shows the empirical cumulative distribution of the relative estimation error for the two networks. Most of the




(a) Network time allocation.

(b) Network allocation by location.

Fig. 4: Experimental results - network allocation for each evaluated scheme, by time and location.


ISBN: 978-3-903176-71-3

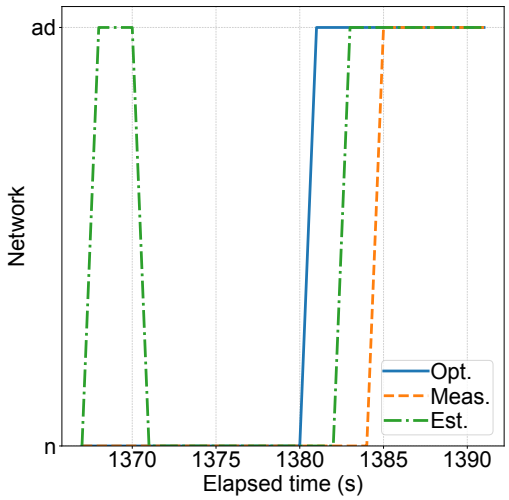



Fig. 5: Experimental results - network choice timeline slice.

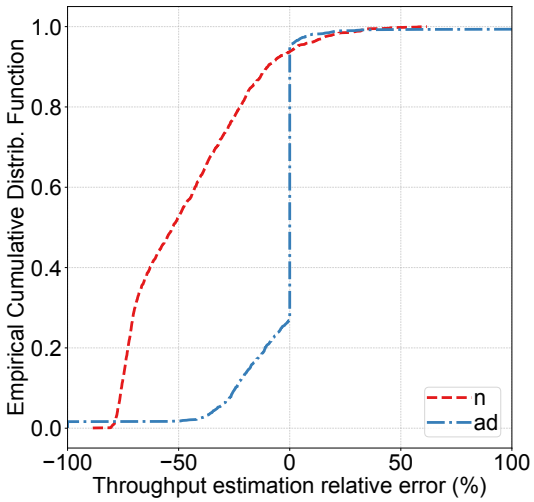



Fig. 6: Experimental results - throughput estimation error.

error stems from an underestimation of the 802.11 n network's performance. This is understandable given that the estimation model was trained on a dataset generated with significantly lower-performing 802.11n hardware.

Ultimately, the close-to-optimal overall performance of the estimate scheme is evidence that, as long as the ranking of network performances is unaffected, it copes with estimation error very well.

The dataset resulting from our experiments is available on Gitlab [11] for further analysis.

## V. Related work

This paper builds upon our earlier work on throughput estimation and forecasting [4], [5], and network selection [5] algorithms, detailed in §II, and is thus most closely related to it. In this section we explore related work by other authors.

One way to reduce the load vehicles place on their cellular connections is to optimize their use. For example, Sliwa *et al.* [12] proposed a scheme that schedules transmission of delay-tolerant traffic for times that avoid channel contention. We pursue the alternative of using public Wi-Fi networks to opportunistically offload data. Multiple experimental studies have shown this to be feasible. The previously mentioned Porto, Portugal study by Aguiar *et al.* [2] confirmed the availability of public hotspots on the majority of streets of the city. The usability of such hotspots was confirmed by a large multi-city study [13], which reported median connection times from 9 s for Paris and Macao, to 22 s for Los Angeles. Furthermore, they found that up to 1 GB/h can be offloaded from a moving vehicle in this manner.

The growing relevance of V2I Wi-Fi communication is highlighted by the recent creation of an automotive topic interest group within the IEEE 802.11 working group [14].

Vehicular data offloading has been studied extensively. Cabernet [15] was pioneering. It features a streamlined network association procedure to maximize transfer time, and a custom transport protocol that improved performance by removing TCP's assumption that losses always mean congestion, which is untrue for wireless environments. In terms of network selection however, it is very simple, connecting to the first open AP it finds, and staying as long as possible. WiChoose could therefore be added to Cabernet to improve its network selection. Wiffler [16] is a system that combines both cellular and Wi-Fi connections, offloading delay-tolerant data to Wi-Fi. Again, it features no particular Wi-Fi network selection scheme, so WiChoose would complement it well.

A more recent work, X-Fi [13], focuses on efficient offloading using provider-managed Wi-Fi networks that require authentication. It uses signal strength as its network selection heuristic. In a diverse scenario featuring multiple Wi-Fi standards and loads, this is insufficient. Work by Giannoulis *et al.* [17], and Deshpande *et al.* [18] proposed using historical data to guide selection, but relied solely on signal quality as a performance indicator. It did not consider mobility.

Energy consumption has also been considered a network selection metric for multi-radio technology environments [19], [20]. That does not make sense for WiChoose, as it focuses specifically on Wi-Fi, and vehicles are not energy constrained.

Ndashimye *et al.* [21] proposed a mobility-aware AP selection scheme. In it the vehicle is given the location and range of surrounding APs, and uses its movement vector to determine when to switch APs in order to remain in communication range. However, it unrealistically assumes all APs provide the same throughput and have fixed circular ranges.

Multiple works [22]–[24] have formulated network selection as a Markov Decision Process (MDP). In an MDP, at each point the agent (in this case vehicle) picks an action: to stay on the current network or switch to a different one. The action takes the system from one state to another, with the end state being probabilistically determined by the previous state and action combination. Each state transition is accompanied by a reward. Solving the MDP equates to finding an action for each possible state, such that the overall reward is maximized.


ISBN: 978-3-903176-71-3

Despite the different formulation, the end goal is similar to that of WiChoose, where throughput is the reward.

The MDP work closest to ours is that of Mushahid *et al.* [24]. Realizing that the state transition probabilities are a priori unknown they used reinforcement learning to solve the MDP, and, realizing the importance of mobility, made location and speed part of the MPD state. However, they did not include direction of movement, a key predictor [5]. They also did not evaluate their scheme experimentally.

## VI. Conclusions

We presented WiChoose, a practical system for online Wi-Fi network selection, and a testing framework for such selection schemes. Our proposed strategy hinges on estimating current network throughput from passively-observable variables. Estimates are clustered by the mobility features observed at the time they were made. The clusters are then used to forecast throughput evolution. Finally, the network forecast to maximize the amount of transferable data is chosen.

We experimentally evaluated this estimation-based scheme with WiChoose, in a realistic-mobility scenario featuring two networks with contrasting ranges and throughputs. It proved very effective, achieving 95 % of the theoretical optimum, and counter-intuitively beating a measurement-based scheme, due to its faster response to changes in channel conditions. This is remarkable given that it doesn't introduce any probe traffic onto the network.

As a testing framework, WiChoose paves the way for future larger-scale experiments with more selection strategies, clients, networks, and mobility patterns. Another interesting future research avenue is to expand the scope of WiChoose to include the issues of network association, authentication, and handover, with the goal of creating a complete Wi-Fi solution for V2I communication.

## Acknowledgment

The authors would like to thank Marta Meireles and Luís Silva for help with the field experiments.

## References

[1] F. para a Ciência e Tecnologia I.P. (FCT), "Project UIDB/50008/2020," 2020. [Online]. Available: https://doi.org/10.54499/UIDB/50008/2020

[2] A. Aguiar and J. Rodrigues, "SenseMyCity: a Mobile IoT Tool for Researching Intelligent Urban Mobility," in *14th Int. Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 2022. DOI: 10.1109/COMSNETS53615.2022.9668516 pp. 725–733.

[3] M. Harris, "The radical scope of Tesla's data hoard," *IEEE Spectrum*, vol. 59, pp. 40–45, 10 2022. [Online]. Available: https://spectrum.ieee.org/tesla-autopilot-data-scope

[4] D. Teixeira, R. Meireles, and A. Aguiar, "Wi-Fi throughput estimation and forecasting for vehicle-to-infrastructure communication," *Computer Communications*, vol. 214, pp. 223–233, 2024. DOI: https://doi.org/10.1016/j.comcom.2023.12.005

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

[6] ——, "A Dataset for Exploring Wi-Fi Network Diversity in Vehicle-to-Infrastructure Communication," 2020. [Online]. Available: https://doi.org/10.5281/zenodo.6884094

[7] D. P. Searson, "GPTIPS 2: An Open-Source Software Platform for Symbolic Data Mining," in *Handbook of Genetic Programming Applications*, A. H. Gandomi, A. H. Alavi, and C. Ryan, Eds. Springer Int. Publishing, 2015, pp. 551–573. ISBN 978-3-319-20883-1

[8] TP-Link, "AD7200 Multi-Band Wi-Fi Router Datasheet," 2018, (accessed on 2024-07-26). [Online]. Available: https://www.tp-link.com/en/home-networking/wifi-router/ad7200

[9] D. Steinmetzer, D. Wegemer, and M. Hollick, "Talon Tools: The Framework for Practical IEEE 802.11ad Research," 2017, (accessed on 2024-07-26). [Online]. Available: https://seemoo.de/talon-tools

[10] "OpenWrt - Open Wireless Router," 2024, (accessed on 2024-06-02). [Online]. Available: https://openwrt.org

[11] R. Meireles, "WiChoose: Wi-Fi network selection for V2I Communication," https://gitlab.com/rui-meireles/wichoose, 2024.

[12] B. Sliwa, R. Adam, and C. Wietfeld, "Client-Based Intelligence for Resource Efficient Vehicular Big Data Transfer in Future 6G Networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5332–5346, 2021. DOI: 10.1109/TVT.2021.3060459

[13] F. Yang, A. Ferlini, D. Aguiari, D. Pesavento, R. Tse, S. Banerjee, G. Xie, and G. Pau, "Revisiting WiFi offloading in the wild for V2I applications," *Computer Networks*, vol. 202, 2022. DOI: 10.1016/j.comnet.2021.108634

[14] IEEE, "IEEE 802.11 Automotive Topic Interest Group," 2024. [Online]. Available: https://www.ieee802.org/11/Reports/auto_update.htm

[15] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: Vehicular Content Delivery Using WiFi," in *Proc. of the 14th ACM Int. Conference on Mobile Computing and Networking*, ser. MobiCom '08. ACM, 2008. DOI: 10.1145/1409944.1409968. ISBN 9781605580968 p. 199–210.

[16] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting Mobile 3G Using WiFi," in *Proc. of the 8th Int. Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. ACM, 2010. DOI: 10.1145/1814433.1814456. ISBN 9781605589855 p. 209–222.

[17] A. Giannoulis, M. Fiore, and E. W. Knightly, "Supporting Vehicular Mobility in Urban Multi-hop Wireless Networks," in *Proc. of the 6th Int. Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '08. ACM, 2008. DOI: 10.1145/1378600.1378608. ISBN 978-1-60558-139-2 pp. 54–66.

[18] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das, "Predictive Methods for Improved Vehicular WiFi Access," in *Proc. of the 7th Int. Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '09. ACM, 2009. DOI: 10.1145/1555816.1555843 pp. 263–276.

[19] I. Joe, W.-T. Kim, and S. Hong, "A Network Selection Algorithm considering Power Consumption in Hybrid Wireless Networks," in *2007 16th Int. Conference on Computer Communications and Networks*, 2007. DOI: 10.1109/ICCCN.2007.4317990 pp. 1240–1243.

[20] H. Liu, C. Maciocco, V. Kesavan, and A. L. Y. Low, "Energy efficient network selection and seamless handovers in Mixed Networks," in *2009 IEEE Int. Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops*, 2009. DOI: 10.1109/WOWMOM.2009.5282451 pp. 1–9.

[21] E. Ndashimye, N. I. Sarkar, and S. K. Ray, "A Novel Network Selection Mechanism for Vehicle-to-Infrastructure Communication," in *2016 IEEE DASC/PiCom/DataCom/CyberSciTech*, 2016. DOI: 10.1109/DASC-PICom-DataCom-CyberSciTec.2016.94 pp. 483–488.

[22] E. Stevens-Navarro, V. W. S. Wong, and Y. Lin, "A Vertical Handoff Decision Algorithm for Heterogeneous Wireless Networks," in *2007 IEEE Wireless Communications and Networking Conference*, 2007. DOI: 10.1109/WCNC.2007.590 pp. 3199–3204.

[23] S. Goudarzi, W. H. Hassan, M. H. Anisi, and S. A. Soleymani, "MDP-Based Network Selection Scheme by Genetic Algorithm and Simulated Annealing for Vertical-Handover in Heterogeneous Wireless Networks," *Wireless Personal Communications*, vol. 92, no. 2, pp. 399–436, 1 2017. DOI: 10.1007/s11277-016-3549-5

[24] M. Hussain, F. França, and A. Aguiar, "Context-aware reinforcement learning for supporting wifi connectivity for vehicles," in *2023 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2023. DOI: 10.1109/VNC57357.2023.10136333 pp. 65–68.


ISBN: 978-3-903176-71-3