Handover Triggering in IEEE 802.11 Networks

Nicolas Montavont, Alberto Blanc, Renzo Efrain Navas, Tanguy Kerdoncuff, German Castignani

Handover Triggering in IEEE 802.11 Networks

Nicolas Montavont, Alberto Blanc, Renzo Navas and Tanguy Kerdoncuff
Institut Mines Telecom / Telecom Bretagne
2, rue de la chataigneraie
35576 Cesson
Email: firsname.lastname@telecom-bretagne.eu

German Castignani
University of Luxembourg / SnT
4, rue Alphonse Weicker
L-2721 Luxembourg
Email: german.castignani@uni.lu

***Abstract*—The current and future IEEE 802.11 deployment could potentially offer wireless Internet connectivity to mobile users. The limited AP radio coverage forces mobile devices to perform frequent handovers while current operating systems lack efficient mechanisms to manage AP transition. Thus we propose an anticipation-based handover solution that uses a Kalman filter to predict the short term evolution of the received power. This mechanism allows a mobile device to proactively start scanning and executing a handover as soon as better APs are available. We implement our mechanism in Android and we show that our solution provides a better wireless connection.**

I. Introduction

Due to the proliferation of Wifi hot-spots and community networks, we have recently observed a great evolution of IEEE 802.11 networks especially in urban scenarios. These 802.11-based networks allow mobile users to get connected to the Internet, providing a high throughput but a limited mobility due to the short coverage area of access points (APs). In our previous work [1] we have shown that community networks appear to be highly dense in urban areas, generally providing several APs (15 in median) per scanning spot. Under this condition, a mobile user may be able to connect to community networks and compensate the low AP coverage area by transiting between APs. We call such AP transition a *handover*. However, two main issues currently limit mobile users from using community networks in such a mobility-aware scenario. First, operators have not deployed the necessary infrastructure to allow mobile users to perform handovers without being disconnected at the application layer, i.e., after a handover on-going application flows are interrupted. This limitation may be addressed by deploying a Mobile IP [2] infrastructure, in which the application flows may be tunnelled through a Home Agent that belongs to the operator. Second, independently from the first issue, there is still a lack of mechanism to intelligently manage a layer 2 handover between two APs. In current mobile devices, when a handover occurs, we observe a degradation of on-going flows corresponding to a dramatic reduction of the TCP congestion window (CWND) and of the throughput. In this paper, we focus on this latter issue by analyzing the impact of layer 2 handovers on mobile users. We propose Kalman-filter-based HAndover Trigger algorithm (*KHAT*) that succeeds in intelligently triggering handovers and reducing the scanning impact on the mobile device. We propose a complete implementation of our handover mechanism in Android ICS (4.0) and show a comparative study to show that our approach outperforms the handover mechanism that is currently implemented on these devices.

978-1-4799-8461-9/15/$31.00

The paper is organized as follows. Section II presents the litterature on handover optimization and Section III analyzes the handover impact on on-going communications. Section IV introduces KHAT which is evaluated indoor and outdoor in Section V. Section VI concludes the paper.

II. Handover Process and Related Work

The IEEE 802.11 standard defines a handover as a three steps process: scanning, authentication and association. The standard proposes two different scanning algorithms namely passive and active scanning. In passive scanning, the mobile station (MS) simply tunes its radio on each channel and listens for periodic *beacons* sent by the APs. In active scanning, the MS proactively sends requests in each channel and waits for responses during a pre-defined timer.

Once candidate APs have been found, the MS selects one of the APs and attempts authentication and association. If the association is successful, the MS can send and receive data through the new AP, if this new AP is on the same IP subnet as the previous AP. If the new AP belongs to another IP subnet, the MS needs additional processing to update its IP address and redirect data flows to its new point of attachment. Such Layer 3 handover may be handled by specific protocols like Mobile IP [2]. Note that in this paper we do not address IP mobility and any layer 3 mobility management protocol can be use on top of our proposal if needed.

In 2012, the IEEE has published new amendments for IEEE 802.11 handover optimization, aimed at reducing its duration and its impact on higher layers. The IEEE 802.11k amendment proposes mechanisms for radio resource measurement for seamless AP transition, including measurement reports of signal strength (RSSI) and load of nearby APs. Additionally, the IEEE 802.11r amendment contains a Fast Basic Service Set Transition (FT), which avoids exchanging 802.1X authentication signaling under special conditions by caching authentication data.

While these features may enhance the handover performance, they heavily rely on a cooperation between APs, which might not always be a viable solution. In addition, users may access various networks operated by different providers. In that case, operators should share network information and performance among them, which is quite an unlikely scenario. In this paper, we focus on MS-based solutions, where the MS itself handles the handover without the help from the network. Several works have been proposed in the literature so far. In general, those studies cover different aspects of the handover mechanism. We may group them into three main categories:

- Handover triggering: when to decide that a disconnection with the current AP will occur.
- AP discovery: how to search for APs on different channels by minimizing the impact on the higher layers.
- Best AP selection: with which AP to associate, among the discovered ones.

The simplest mechanism to trigger a handover is to monitor the RSSI as an estimation of the link quality and start the handover process if the current RSSI is lower than a pre-established threshold (commonly set at $-80\,dBm$). Fig. 1a shows the relationship between the RSSI measured on an MS and the TCP throughput that we have gathered during more than 600 connections to community networks in a urban area in Rennes, France [1]. We observe that the TCP throughput is extremely variable for high RSSI, but starts degrading for RSSI lower than $-70\,dBm$, and it becomes significantly low around $-80, dBm$.

Some works focus on the anticipation of the handover triggering in order to minimize the impact on ongoing communications. Mhatre et al. [3] propose a set of handover algorithms based on continuously monitoring the wireless link, i.e., listening to beacons from the current and neighboring channels. These approaches give handover latencies varying between 150 and 800 ms. However, since these approaches need to listen to beacons from neighboring channels, it is necessary to modify the firmware of the wireless card, which may not always be possible. Yoo et al. [4] propose a number of handover triggering mechanisms based on predicting RSSI samples at a given future time using Least Mean Square (LMS) linear estimation. In this algorithm, the device continuously monitors the RSSI and computes the LMS prediction if the RSSI is below a certain threshold (P_{Pred}). Then, if the predicted RSSI value is lower than a second threshold, P_{Min}, the MS starts a handover.

Wu et al. [5] propose a handover mechanism aiming at decoupling the AP discovery phase from the AP selection and reconnection phase. The MS alternates between scanning phases and a (normal) data mode where the MS is connected to its current AP. The time interval between two scanning phases is adapted depending on the current signal level and varies between 100 and 300 ms. In each scanning phase, the sequence of channels to scan is selected based on a priority list that is built based on the results of a periodic full scanning (i.e., here all channels are scanned).

As far as Android devices are concerned, Silva et al. [6] present a mobility management solution based on IEEE 802.21. They propose a mapping of IEEE 802.21 primitives for handover initiation, preparation, execution and completion to existent Android OS methods and functions.

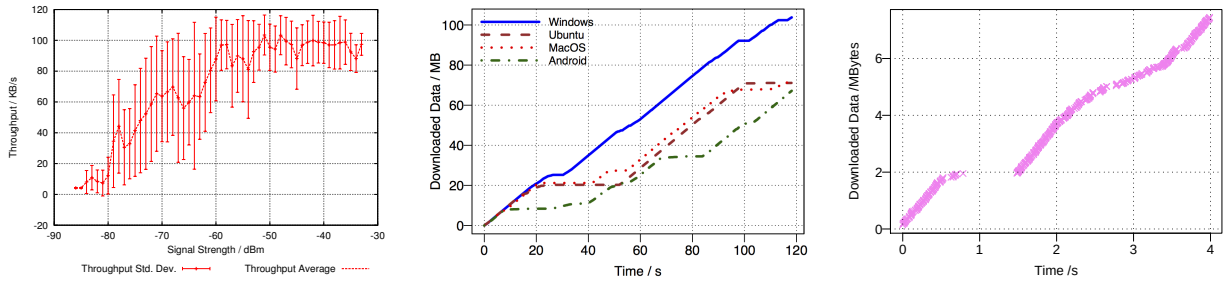
III. Handover Impact

During an L2 handover, the MS is not able to send or receive application flows. This is because, usually, when a MS triggers a handover, the link quality does not allow exchanging frames anymore, and because the MS is often switching operating channel. In this section we evaluate the handover and scanning impact on application flows, and determine which parameters influence the scanning latency and success rate.

This testbed consists of nine Cisco Aironet 1040 APs installed in the roof of our building at the locations given in Fig. 2. All APs are connected to a dedicated wired LAN. APs broadcast a single SSID, corresponding to an open-system authentication network belonging to a single IP subnet. We also use a dedicated (fixed) server for traffic generation and tracing. iPerf is used to generate TCP downlink traffic to the MS. For each experiment, we walk from AP_1 to AP_6 and then back again to AP_1.

A. Operating Systems Benchmark

To illustrate how the handover is currently impacting data flows, we have performed a set of experiments to evaluate the degradation of TCP performance for different devices and Operating Systems (OS). Table I shows the number of handovers and the average TCP



(a) RSSI and TCP Throughput Relation (b) Downloaded data for different OS (c) Scanning impact on TCP download

Fig. 1: Various TCP performance

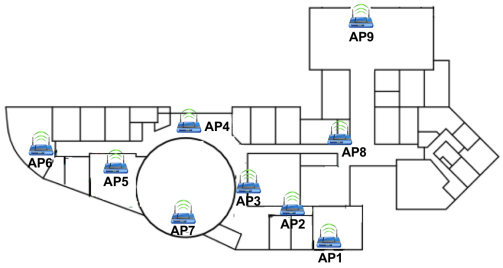


Fig. 2: Campus AP Deployment

throughput we have observed for the same path and same MS speed using different devices and operating systems. As a baseline, we also show the maximum achieved throughput for each device remaining static and connected to a single AP. Using Windows, we observe the best result, since the MS performs up to four handovers, reaching an average throughput of $0.875\,MB/s$. Additionally we observe that for Windows, the time in which no data is downloaded (i.e., the disconnected time) is relatively short compared to the other OSs. The netbook running Ubuntu reacts slowly to changing channel conditions: in this case the MS is disconnected for more than 20 s and executes only two handovers, indicating that the MS waits until the quality of the radio link is significantly degraded. Fig.1b shows the evolution of the downloaded data for each case. Additionally, we have observed that for the Windows device, the average round-trip time (RTT) is the lowest one (103 ms) having also a low standard deviation. This differs from the other devices that reach larger RTT values.

B. Scanning Interactions with Data Traffic

We focus on active scanning where an MS sends Probe Requests on each channel to discover potential APs, instead of just waiting for periodic beacons (passive scanning). We chose active scanning because it allows spending less time in each channel to determine the AP availability. If the handover phases are done one after the other, all packets that arrive during the handover process will be lost. In order to reduce the impact of handovers on applications flows, it is possible to introduce a gap between the scanning phase and the other handover steps, i.e., the decision to handover, the authentication and the association, as presented in [5]. An MS may use the power saving mode defined in IEEE 802.11 to request its current AP to buffer incoming packets during the time the MS scans other channels. This way, instead of loosing packets during the scanning phase, an MS can receive the packets after the scanning phase, albeit with an extra delay. This behavior is illustrated in Fig. 1c, where we plot the sequence number of the received packets of a TCP flow when an MS is performing one scan of the 13 channels with an active timer set at $50\,ms$. We can see that the scan is starting just before the time $1\,s$, at which no more data packets are received from the server. Once the scan is finished, around $850\,ms$ after, the MS comes back to its current AP, and starts receiving TCP packets again.

This technique can also be used to split a scanning phase into several sub-phases where only a subset of channels are scanned. For example, to scan the 13 channels, an MS could sequentially scan three times a subset of 4 (or 5) channels each time, interleaving these sub-phases with the data mode with the current AP to retrieve data packets. The impact of the number of scanned channels, and the timers used in each channel is given in the next subsection.

C. Scanning Parameters

We analyze the scanning performance under different values of timers used to wait for Probe Responses (from 5 ms to 100 ms) and different number of scanned channels during a sub-phase (between 1 and 13). In the standard IEEE 802.11 scanning algorithm, the MS is supposed to scan each channel using two timers

Device	OS Version	Chipset	Static Avg. Thr. (MB/s)	Mob. Avg. Thr. (MB/s)	Number of handovers	Mean RTT (ms)	σRTT (ms)
Asus N10J	Win XP SP2	AR5006	5.19	0.878	4	103	43
Asus N10J	Ubuntu 10.04	AR5006	4.88	0.601	2	161	360
Nexus S	Android 4.0.3	BCM4329	3.80	0.568	5	129	114
MacBook	MAC OS 10.7.4	BCM4322	8.44	0.613	3	167	276

TABLE I: Handover performance of different OS

Nb. of channels	AT=5 (%)	AT=10 (%)	AT=20 (%)	AT=50 (%)	AT=100 (%)
1	3.11	5.76	10.62	22.28	25.24
3	6.45	18.28	32.61	58.18	88.24
5	9.28	21.02	38.83	68.94	89.31
8	10.44	23.61	40.46	70.43	96.58
13	11.74	28.62	45.76	79.88	100.00
RSSI	-67.16	-70.07	-76.02	-81.28	-83.26

TABLE II: Percentage of discovered APs for different values of AT and number of scanned channels

namely MinCT and MaxCT (see section II). However, the IEEE 802.11 Android driver uses a single timer, namely *Active Timer* (AT) for scanning. AT is defined as the time an MS waits for Probe Responses on a channel.

We ran 60 scanning sub-phases for each AT and subset of scanned channels and measured the average number of discovered APs, the RSSI distribution of the discovered APs and the average duration of the scanning (i.e., the scanning latency). Results are presented in Table II. As a baseline, we consider that all the available APs are discovered when scanning the full channel sequence (i.e., 13 channels) using AT=100 ms. In the other cases, the MS discovers only a fraction of the APs, since it either does not wait long enough to receive all AP Probe Responses, or because only a subset of channels are scanned.

We have also observed that when using a short AT, even if the MS discovers a low number of APs, those APs have a high RSSI. On the other hand, when using higher AT values, the MS discovers more APs but a large part of them have a low RSSI. This can be observed in Fig. 3a, where we see that for AT=5 ms the average RSSI of candidate APs is -67 dBm, while for AT=20 ms, this decreases up to -76 dBm.

IV. KHAT: Proactive Handover Algorithm

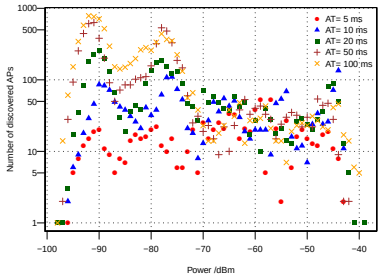
We propose a handover algorithm called Kalman Filter-based HAndover Triggering (KHAT for short) that provides link going down detection, optmized scanning strategy, and new AP selection. An MS monitors its link quality with its current AP, and when the signal strength is degrading, it starts alternating between scan periods and data communication with the current AP. The scan periodicity and the timer values are determined according to the current link quality and whether a candidate AP has already been found. Once the candidate AP becomes better than the current AP, the handover is triggered.

A. RSSI modelling

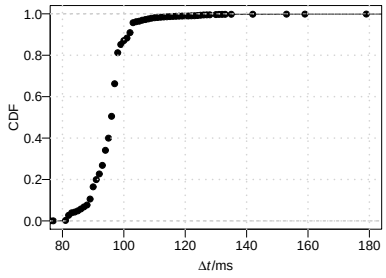
One way of keeping track of the changing radio condition is to track the RSSI on the MS. While far from being perfect, the RSSI has the advantage of being always available, whether the MS is exchanging data or not, as it is updated not only whenever the MS receives data frames but also when it receives beacon frames, which are typically sent every 100 ms by most APs. As the RSSI can fluctuate rapidly, especially when a user is moving, its instantaneous value is not necessarily representative. At the same time, its local average and trend are more useful in deciding whether the radio channel conditions are improving or not and whether they are reaching the point where communication is no longer possible. Using the well known Kalman filter, it is possible to extract this information from the RSSI measurements. Many authors have already use Kalman filter and other time series techniques in order to model radio channels and the received signal strength, see, for example, the works by Jiang et al. [7], by Baddour et al. [8] and references therein.

More formally, let $X(t_i) \triangleq X_i$ be the received signal strength at time t_i. In our case, we sample the RSSI roughly every 100 ms; but, as we rely on software timers, there are no guarantees that the t_i's will be equally spaced. Figure 3b shows the empirical distribution of $\Delta t_i = t_i - t_{i-1}$ for a subset of the traces we collected. The average is 96 ms and the standard deviation is 8.2 ms. Given that roughly 90% of the samples are within less than 100 ms of each other, it seems reasonable to "re-sample" the time series with a time-step of 100 ms.

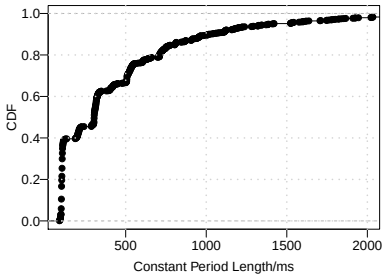
In all the traces we have collected, it is often the case that several consecutive samples have the same value, indicating that the received signal strength is often constant during periods that are longer than the average distance between samples. The presence of several samples with exactly the same value is an obstacle when one is trying to estimate the local trend of a signal as, in this case, the estimated slope would be exactly 0. The Kalman filter does not perform well in these circumstances. As we rely on the values reported by the 802.11 driver, we wondered whether these consecutive samples with the same values were caused by the driver



(a) Number and RSSI of discovered APs for different AT



(b) Sampling interval CDF for the original time series



(c) The CDF of the length of the periods where all the received power samples are equal for the community networks

Fig. 3: RSSI analysis

not updating the values often enough. Figure 3c shows the distribution of the length of the periods where the signal strength was constant for several traces collected using a static MS which was not sending or receiving data (note that the distribution was the same whether the test was performed using a laptop running Linux or a smartphone running Android). The median is 305 ms and the standard deviation is 306 ms. In the case of mobile MSs and/or data traffic the median values are smaller (around 110 ms for MS with data traffic, but the standard deviation is always larger). In order to mitigate the effect of these periods, we pre-process the RSSI samples, using a time-varying exponential average, before applying the Kalman filter. In order to further reduce the lag of the smoothed signal, we use a time-varying weight in the exponential smoothing.

Let Y_i be the re-sampled RSSI time series. We construct the smoothed series Z_i as:

$$Z_i = \alpha_i Y_i + (1 - \alpha_i) Z_{i-1}$$

where $Z_1 = Y_1$ and $\alpha_i = \alpha_{\text{up}}$ if the RSSI is increasing, instead whenever the RSSI starts decreasing $\alpha_i = \alpha_1$. Whenever the RSSI is constant the value of α_i is determined by the last change before the beginning of the constant samples. If the last change was an increase $\alpha_i = \alpha_{\text{up}}$, otherwise $\alpha_i = \min(0.8 \cdot \alpha_{i-1}, \alpha_{\text{min}})$. This corresponds to the pseudo-code in Algorithm 1.

We have used $\alpha_{\text{up}} = 0.5$, so that the smoothed time series will react quickly to upward changes, $\alpha_1 = 0.4$ and $\alpha_{\text{min}} = 0.01$ so that it will, instead, react much more slowly to downward changes. The reason of this asymmetric behavior is that we are interested in having an accurate estimate of the level and, above all, of the trend, only when the signal is decreasing. By using a larger α_i when the signal is increasing we assure that Z will quickly reach the value of Y reducing the lag between Y and Z.

Algorithm 1 The algorithm used to compute α_i

```
increasing ← FALSE
lastValue ← Y_1
α_1 ← α_1
i ← 1
while i ≤ length(Y) do
    if Y_i ≠ lastValue then
        if Y_I > lastValue then
            increasing ← TRUE
            α_i ← α_up
        else
            increasing ← FALSE
            α_i ← α_1
        end if
    else
        if increasing =FALSE then
            if α_{i-1} > α_min then
                α_i ← 0.8 · α_{i-1}
            else
                α_i ← α_{i-1}
            end if
        else
            α_i ← α_up
        end if
    end if
end while
```

We have verified that the received power times series of our sample are indeed non-stationary by computing their autocorrelation functions, which were all slowly decreasing (as a function of the lag). We have then decided to use a state-space based model to represent the evolution of the power over time. In particular we have used the local linear trend model (see, for example, Durbin and Koopman [9]):

$$\begin{aligned} Z_i &= \mu_i + \varepsilon_i & \varepsilon_i &\sim \mathcal{N}(0, \sigma_\varepsilon^2) \\ \mu_{i+1} &= \mu_i + \nu_i + \xi_i & \xi_i &\sim \mathcal{N}(0, \sigma_\xi^2) \\ \nu_{i+1} &= \nu_i + \zeta_i & \zeta_i &\sim \mathcal{N}(0, \sigma_\zeta^2) \end{aligned} \quad (1)$$

where Z_i is the time series under scrutiny, μ_i is the level at time i, ν_i is the slope at time i and $\varepsilon_i, \xi_i, \zeta_i$ are independent identically distributed (i.i.d.) Gaussian random variables with 0 mean and variance $\sigma_\varepsilon^2, \sigma_\xi^2, \sigma_\zeta^2$ respectively. These variances can be obtained by Maximum Likelihood Estimation from sample realizations of Z.

Once the values for the variances are specified, and given a realization of Z_i ($i = 0, \ldots, n$), one can use the well known Kalman filter algorithm to compute μ_i and ν_i for any value of i (again, see, for example, Durbin and Koopman [9]). To be more precise, one can solve the "filtering" problem, where the values of μ_i and ν_i are computed using the samples $Z_0, Z_1, \ldots, Z_i$. At first, we have used the `dlm` [10] R [11] package to solve the filtering problem. Note that, as the filtering problem uses only the samples between 0 and i, it can be implemented in real time as it depends only on past values of the time series. The Kalman filter can also be used to predict future values. In the case of the local linear trend model (1), the prediction algorithm is extremely simple: one can just model the future values of the time series using a straight line with slope ν_i, starting at the value μ_i at time i.

We have also implemented the Kalman filter on a Samsung Nexus S smartphone, in the WiFiStateMachine module of the Android Java framework. For the sake of simplicity we have used a straightforward implementation of the Kalman recursion. The general form of the Kalman filter is:

$$\begin{aligned} Z_i &= F\Theta_i + v & v_i &\sim \mathcal{N}_n(0, V_i) \\ \Theta_i &= G\Theta_{i-1} + w_i & w_i &\sim \mathcal{N}_m(0, W_i) \end{aligned}$$

For the local linear trend model Z is a scalar, and so is v, while:

$$\Theta_i = \begin{pmatrix} \mu_i \\ \nu_i \end{pmatrix}, G = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, W = \begin{pmatrix} \sigma_\xi^2 & 0 \\ 0 & \sigma_\zeta^2 \end{pmatrix}, F = (1 \quad 0)$$

We are interested in computing the 2×2 vector $m_i = (\mathrm{E}[\mu_i]\, \mathrm{E}[\beta_i])^T$, containing the expected values of the level (μ_i) and slope (ν_i). It is known [9] that one can compute these values using the following equations:

$$\begin{aligned} m_i &= a_i + R_i F_i^T Q_i^{-1} e_i & f_i &= F_i a_i \\ C_i &= R_i - R_i F_i^T Q_i^{-1} F_i R_i & Q_t &= F_i R_i F_i^T + V_i \\ a_i &= G_i m_{i-1} \\ R_i &= G_i C_{i-1} G_i^T + W_i \end{aligned}$$

where $e_i = Y_i - f_i$, and the following initial values: $C_0 = \left(\begin{smallmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{smallmatrix}\right)$, $m_0 = (Z_0 \quad 0)^T$ are used for C and m. The values of σ_1^2 and σ_2^2 have almost no influence on the computations as the matrices R, F, T, C quickly converge to steady state values which are independent from the initial values.

The local linear trend model 1 is characterized by three parameters: $\sigma_\varepsilon^2, \sigma_\xi^2, \sigma_\zeta^2$. It is possible to use Maximum Likelihood Estimation (MLE) methods to estimate the values of these parameters from sample realizations of Z. We have used the MLE functions of the `dlm` package to this end, but in some cases the optimization algorithm used to compute the MLE did not converge. When it converged its estimations for σ_ε^2 and σ_ζ^2 where not always consistent over all the samples but the order of magnitude was fairly consistent, with σ_ε^2 usually smaller than σ_ζ^2 and with σ_ε^2 often fairly close to 0. It should be stressed that, in this case, there are no guarantees about the convexity of the optimization problem solved by the MLE procedure, which can very well converge to a local minimum instead of a global one. Also it is not uncommon to tune the model parameters in order to improve its performances. In our case we have observed that using $\sigma_\varepsilon^2 = 0.5$, $\sigma_\xi^2 = 1$ and $\sigma_\zeta^2 = 2.5$, we obtain fairly smooth level and slope values, which can be effectively used by the KHAT algorithm.

B. Algorithm design

KHAT adapts the scanning strategy, the scanning period and the handover trigger by comparing an estimate of the link quality and the quality of candidate AP as presented in Fig. 4. The main process consists in continuously monitoring the RSSI of the current link and detect a link going down event. To achieve this, we use the Kalman filter to obtain the current value of the RSSI (μ) and the slope (ν). After analyzing a large number of RSSI time series, we have estimated that the link going down trigger can be declared if $\mu < -70\,\mathrm{dBm}$ and $\nu < -0.2\,\mathrm{dBm/s}$. If the link going down condition is satisfied, the MS check on its candidate AP list. If there is not a valid candidate AP, the MS will attempt scanning only if there has not been another scanning instance for the last T_{Scan} seconds. On the other hand, if after triggering a link going down condition, the MS has a valid candidate it will attempt a handover only if the difference between the candidate AP RSSI and the current exponentially smoothed RSSI sample (μ) is greater than Δ, where Δ is defined as follows:

$$\Delta = \begin{cases} 8 & \text{, if } \mu > -70 \\ 5 & \text{, if } -70 \leq \mu < -75 \\ 3 & \text{, if } -75 \leq \mu < -80 \\ 2 & \text{, if } \mu \leq -80. \end{cases} \tag{2}$$

After a scan completes, an existing candidate AP would be updated with a new RSSI, or a new candidate may be selected. Additionally, in order to avoid

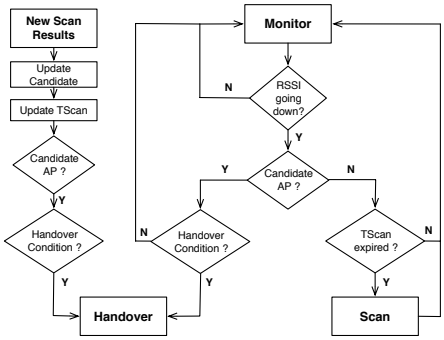


Fig. 4: Algorithm Flow Chart

μ (dBm)	Number of channels	Active Time (ms)	Scanning duration (ms)
$(+\infty, -75)$	13	5	150
$[-75, -80)$	13	10	250
$[-80, -\infty)$	13	20	400

TABLE III: Scanning Strategies

scanning at a very high frequency, we adapt the value of T_{Scan} depending on the scanning results. Each time the MS triggers a link going down, if a candidate AP exists, we double the current value of T_{Scan} (up to 1 s) since it is not necessary to scan at a high frequency if there is at least one candidate AP. On the other hand, if no candidate exists at that time, we set T_{Scan} to its minimum value (250 ms).

The scanning strategy itself is also adapted depending on the current link condition. Each scanning strategy consists in determining a number of channels to scan and the time to wait on each channel (AT in Android system). Based on results presented in section III-C we fixed AT as presented in Table III: the better the current link quality is, the less time the MS will spend scanning, because it still has time to find APs before it disconnects from its current AP. When the signal quality with the current AP is low, we set aside more time for the MS to scan, in order to maximize the probability to find an AP. In order to contain the scanning duration, we propose to use AT in $\{5\,\text{ms}, 10\,\text{ms}, 20\,\text{ms}\}$. The reason is that for smaller scan times, we only find APs with high RSSI (as shown in section III-C) and as we are in fairly good condition, the MS would only be interested in AP with high RSSI.

V. Experimentation

A. Methodology and implementation

We have implemented our solution on the Android ICS 4.0.3 system working on a Samsung Nexus S (GT-I9023) smartphone. It involves modifications in the Android Java Framework, the WPA Supplicant

		KHAT	Stock Android
Indoor	Average Power (dBm)	-63	-68
	Average Throughput (MB/s)	2.11	0.80
Outdoor	Average Power (dBm)	-71	-77
	Average Throughput (MB/s)	0.22	0.12

TABLE IV: Performance comparison

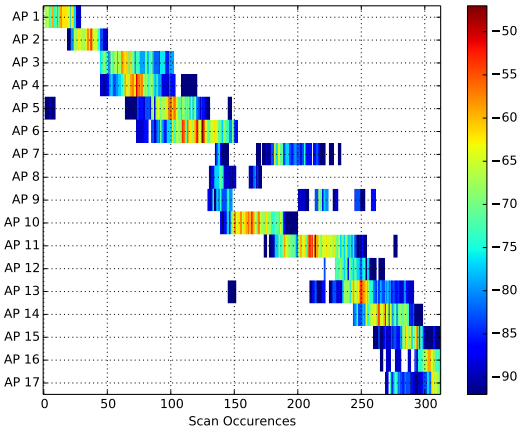


Fig. 9: Heatmap of the selected AP in the outdoor environment

and the Linux driver (BCM4329). We moved at the same time two identical smartphones, one with KATH implementation and the other with stock Android, in two different environments. The first set of experiments was performed in our building as described in Fig 2. The mobile user walks at a roughly constant speed and each connection lasts for 120 s. The second set of experiments was performed in the city of Luxembourg, using the HOTCITY Wifi deployment (see [12] for more details). In all cases, we use iperf to generate the TCP traffic for both MSs and generate several connections for more than one hour.

B. General results

Fig. 5, 6, 7 and 8 show the RSSI and the received TCP data for the two considered environments, over one connection duration, while Table IV shows the average over all connections that we made. We can see that KHAT provides a better RSSI (-69 dBm in average) along the connections and allows the smartphone to have a better throughput than stock Android (222 kB/s versus 146 kB/s). We can also see that KHAT triggers the handover systematically before stock Android to avoid suffering from a poor quality with its current AP. Sometimes, as at Time=400s of the outdoor connection, KHAT manages to find an intermediate AP between those chosen by Stock Android which allows to signigicantly increase both the RSSI and the TCP download. Looking at the zoom of Fig. 8 between Time=200s and

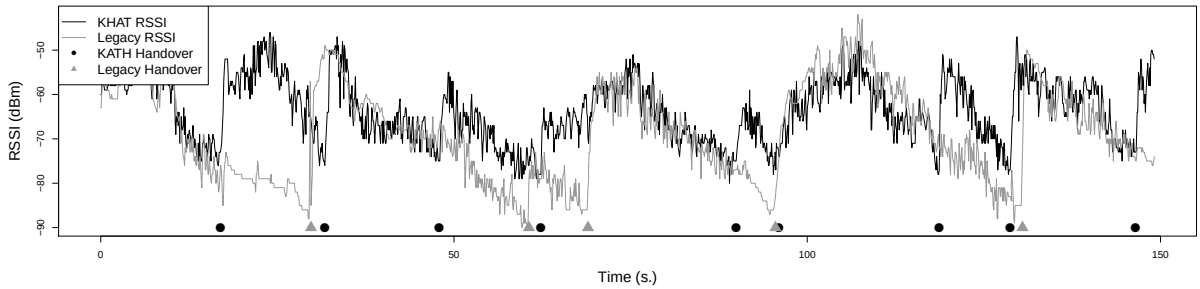


Fig. 5: RSSI for Legacy and KHAT smartphones indoor

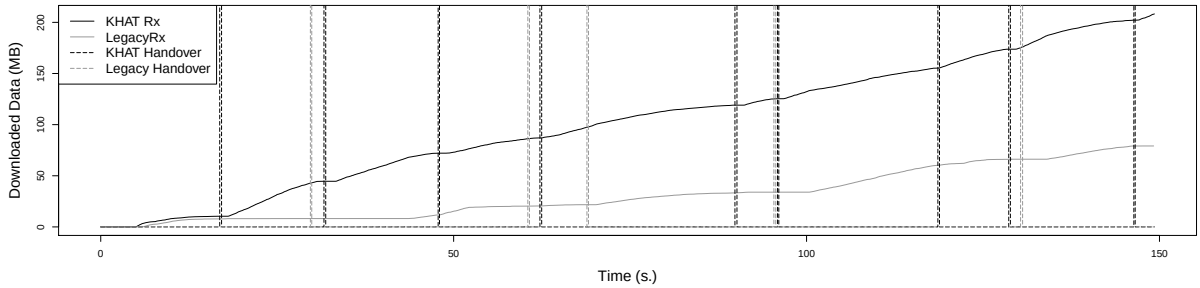


Fig. 6: Received data for Legacy and KHAT smartphones indoor

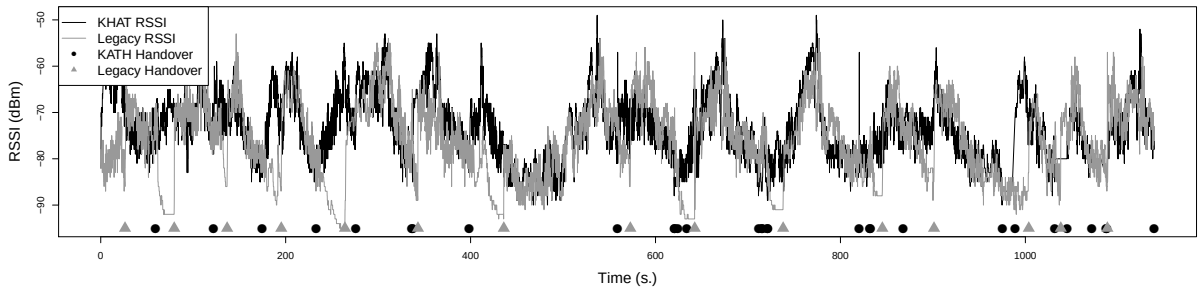


Fig. 7: RSSI for Legacy and KHAT smartphones outdoor

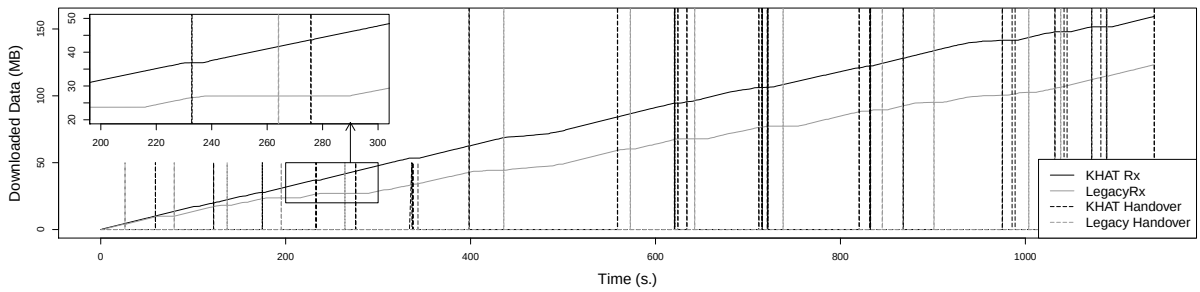


Fig. 8: Received data for Legacy and KHAT smartphones outdoor

300s, we can see a quite large period of time (around $50s$) where the Legacy smartphone is not able to receive any data from the TCP server. On the other hand, KHAT is performing two handovers. The first handover at Time=233s is made prior to the Legacy handover, but still a stagnation in the received data is observed before

and after the handover. However the second handover at Time=275s is smooth and does not impact the data reception.

Fig. 9 shows the list of selected APs from the chosen connection in the outdoor environment captured by an MS running Wi2me[1]. The APs are shown in their apparence order along the path. We can see that for the six first APs, the KHAT AP selection is judicious: when the signal strength of one AP is degrading, another AP becomes available. However, between scanning occurence 140 to 200, there is not ideal choice for any AP. Observing that the RSSI is low, KHAT is trying to handover to different AP at this period of time, osciatilling between AP_7, AP_8, AP_9 and AP_{10}. These are the handovers we can see around Time=625s in Fig 7. While KHAT avoids to trigger handover when it is not for a significantly better AP, in areas where all APs offer low RSSI, KHAT may trigger several handovers. It finally finds AP_{11} at Time=721s which was providing a good coverage area. During this period, we can see that the Legacy phone did not trigger any handover and was unable to receive any data for a long period of time.

VI. Conclusion

IEEE 802.11 is one of the most popular wireless standard to offer high data rate Internet connection. With the vast number of hot-spots and community networks that are deployed today, there is a potential for users to use Wifi network in mobility scenarios. However, as the AP coverage area is usually limited to few tenths of meters, there is a strong need for optimized mobility support, when users move from one AP to another one. We have shown in this paper that current devices are able to transit between APs, but the handover performance is quite low.

We proposed a handover algorithm called KHAT that anticipates the signal loss from the current AP to preemptively scan for potential APs. The prediction of the link going down is achieved with a Kalman filter which estimates the slope of the RSSI to determine the link condition. If the estimate is below a given threshold (smooth RSSI lower than -70 dBm and slope lower than 0.2), we launch a scanning. Data packets can be buffered (and retrieved later on) by the AP during the MS scanning by exploiting the power saving mode defined in 802.11. Depending on the scanning results, the MS will either handover to a new (better) candidate AP that has been found, or it will loop on the link quality prediction. The scanning period and strategy are adapted depending on the current link condition.

We have implemented KHAT on Android ICS 4.0.3 system working on a Samsung Nexus S (GT-I9023). To address the tradeoff between the scanning latency and the AP discovery, the MS is scanning with AT=20 ms if a handover is imminent, AT=10 ms when the link quality is medium and AT=5 ms when the link quality is good. In two different environments (indoor and outdoor), we compared a Stock Android with a KHAT smartphone. We have shown that KHAT outperforms Stock Android by anticipating handovers, and using more APs available on the path. The average RSSI is 6dBm either in the outdoor environment, and the TCP throughout is 0.22MB/s compared to 0.12MB/s for Stock Android. The perspective of this work is to apply the link quality prediction on candidate APs in order to better choose the target AP when a handover is needed.

VII. Acknowledgments

This work has received a French government support under reference ANR-10-LABX-07-01 (Cominlabs).

References

[1] G. Castignani, A. Lampropulos, A. Blanc, and N. Montavont, "Wi2Me: A Mobile Sensing Platform for Wireless Heterogeneous Networks," in *International Workshop on Sensing, Networking and Computing with Smartphones (Phonecom), IEEE ICDCS*, june 2012.

[2] C. Perkins, "IP Mobility Support for IPv4," Internet Requests for Comments, IETF, RFC 3220, february 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3220.txt

[3] V. Mhatre and K. Papagiannaki, "Using smart triggers for improved user performance in 802.11 wireless networks," in *Proc. of the 4th int. conf. on Mobile systems, applications and services*, 2006, pp. 246–259.

[4] S. Yoo, D. Cypher, and N. Golmie, "LMS predictive link triggering for seamless handovers in heterogeneous wireless networks," in *IEEE Military Communications Conference, 2007. MILCOM 2007.* IEEE, Oct. 2007, pp. 1–7.

[5] H. Wu, K. Tan, Y. Zhang, and Q. Zhang, "Proactive scan: Fast handoff with smart triggers for 802.11 wireless LAN," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, May 2007, pp. 749–757.

[6] R. Silva, P. Carvalho, P. Sousa, and P. Neves, "Enabling heterogeneous mobility in android devices," *Mob. Netw. Appl.*, vol. 16, no. 4, pp. 518–528, Aug. 2011. [Online]. Available: http://dx.doi.org/10.1007/s11036-011-0322-6

[7] T. Jiang, N. Sidiropoulos, and G. Giannakis, "Kalman filtering for power estimation in mobile communications," *Wireless Comm., IEEE Trans. on*, Jan 2003.

[8] K. Baddour and N. Beaulieu, "Autoregressive modeling for fading channel simulation," *Wireless Comm., IEEE Trans. on*, july 2005.

[9] J. Durbin and S. Koopman, *Time series analysis by state space methods*, ser. Oxford statistical science series. Oxford Univ. Press, 2001.

[10] G. Petris, "An R package for dynamic linear models," *Journal of Statistical Software*, 2010. [Online]. Available: http://www.jstatsoft.org/v36/i12/

[11] R Development Core Team, *R: A Language and Environment for Statistical Computing*. [Online]. Available: http://www.R-project.org

[12] G. Castignani, J. Monetti, N. Montavont, A. Arcia-Moret, R. Frank, and T. Engel, "A study of urban ieee 802.11 hotspot networks: towards a community access network," in *Wireless Days (WD), 2013 IFIP*, Nov 2013, pp. 1–8.