ComPoScan: Adaptive Scanning for Efficient Concurrent Communications and Positioning with 802.11

Thomas King
Department of Computer Science
University of Mannheim, Germany
king@informatik.uni-mannheim.de

Mikkel Baun Kjærgaard
Department of Computer Science
University of Aarhus, Denmark
mikkelbk@daimi.au.dk

ABSTRACT

Using 802.11 concurrently for communications and positioning is problematic, especially if location-based services (e.g., indoor navigation) are concurrently executed with real-time applications (e.g., VoIP, video conferencing). Periodical scanning for measuring the signal strength interrupts the data flow. Reducing the scan frequency is no option because it hurts the position accuracy. For this reason, we need an adaptive technique to mitigate this problem.

This work proposes ComPoScan which, based on movement detection, adaptively switches between light-weight monitor sniffing and invasive active scanning to allow positioning and to minimize the impact on the data flow. The system is configurable to realize different trade-offs between position accuracy and the level of communication interruption.

We provide extensive experimental results by emulation on data collected at several sites and by validation in several real-world deployments. Results from the emulation show that the system can realize different trade-offs by changing parameters. Furthermore, the emulation shows that the system works independently of the environment, the network card, the signal strength measurement technology, and number and placement of access points. We also show that ComPoScan does not harm the positioning accuracy of a positioning system. By validation in several real-world deployments, we provided evidence for that the real system works as predicted by the emulation. In addition, we provide results for ComPoScan’s impact on communication where it increases throughput by a factor of 122, decreases the delay by a factor of ten, and the percentage of dropped packages by 73 percent.

Categories and Subject Description: C.2.1 [Computer-Communication Networks]: Network Architecture and Design, I.6.7 [Simulation and Modelling]: Simulation Support Systems

General Terms: Experimentation, Measurement

Keywords: movement detection system, positioning system, 802.11, monitor sniffing, communication

1. INTRODUCTION

Back in 1999, when IEEE 802.11 was being standardized, the researchers and engineers working on the standard probably never thought about the new ways we use this technology today. Real-time applications such as voice over IP and video conferencing were a rarity years ago but are a common phenomenon nowadays. These real-time applications have hard requirements in terms of bandwidth, delay, and packet loss to be functional. An even more extreme new way of usage is to utilize the signal strength measurement capabilities of 802.11 network cards as a basis for indoor positioning systems to enable location-based services. Initially, signal strength measurements are performed during a so-called active scan to let a network card detect which access points might be the best to connect to. Many indoor positioning systems (e.g., [2, 6]) make use of 802.11, because almost all modern cell phones and laptops are equipped with this wireless technology. Therefore, the devices can be used for positioning as they come out of the box, which means that no additional hardware is required.

Even the newer sub-standard 802.11b and 802.11g do not satisfy all these requirements. Furthermore, many workarounds and novel approaches (e.g., [5, 13, 14]) have been proposed to make 802.11 ready for many of these new demands. However, still unsolved remains the problem that occurs when a 802.11 network card is utilized for positioning and communicating at the same time. On the one hand, the positioning system requires a steady stream of active scans to be able to deliver accurate position estimates to location-based services. Especially, if the positioning system is used to track users as e.g., required for indoor navigation systems in huge buildings. Performing an active scan means that the network card switches through all the different channels in search of access points. Dependent on the network card, this takes about 600 milliseconds. During this time no communication is feasible. On the other hand, there are the demanding real-time applications. For instance, a video conference requires around 512 KBit/s of bandwidth and a round trip delay of less than 200 milliseconds, depending on the video and voice quality [17].

Figure 1 depicts what happens to throughput and delay of a 802.11g-enabled mobile device if the network card is requested to perform an active scan every 600 milliseconds. During the first 20 seconds communication is untroubled, which means a throughput of about 20 MBit/s on average and that a round trip delay of less than 45 milliseconds is achievable. In the 20th second active scanning kicks in. The remaining seconds only provide 0.1 MBit/s of throughput and 532 milliseconds of delay, because active scans are performed so often. Due to variations in the execution time of scans, on some rare occasions no data transmission is possible at all.
In this paper, we propose a novel solution to this problem which is called ComPoScan. It is based on movement detection to switch, on the basis of adaptability, between lightweight monitor sniffing and invasive active scanning. Only in case that the system detects movement of the user, active scans are performed to provide the positioning system with the signal strength measurements it needs. If the system detects that the user is standing still, it switches to monitor sniffing to allow communications to be uninterrupted.

Monitor sniffing is a novel scanning technique proposed in [8]. It works with most 802.11 network cards around today. Monitor sniffing allows a mobile device to recognize access points operating on channels close to the one it is using for communications with the access point it is associated with. It has been shown that up to seven channels can be overheard without any disturbance of the actual communication.

Our movement detection approach is also based on signal strength measurements. However, the measurements provided by monitor sniffing are sufficient to detect reliably whether the user is moving or standing still. We designed the movement detection system to be configurable so that, depending on the user’s preferences, communication capabilities or positioning accuracy can be favored.

We make the following contributions in this work: First of all, we are the first who present a system to mitigate the effect of scanning on concurrent communications. Secondly, we are the first utilizing monitor sniffing and active scanning to build a reliable indoor movement detection system. Thirdly, we provide a deep investigation by means of emulation to show that our movement detection system works independently of the environment, the network card, the signal strength measurement technology, and number and placement of access points. Additionally, we show that it does not harm the positioning accuracy of the positioning system. Fourthly, we implement ComPoScan and use this prototype in a real-world deployment to gather results showing that the real system works as predicted by the emulation. The results show that our goal of mitigating the effect of scanning on communications is full-filled.

The remainder of this paper is structured as follows: In Section 2, we present the relevant related work. Subsequently, we introduce our novel ComPoScan system. The details of our movement detection approach are discussed and evaluated by means of emulation in Section 4. Section 5 discusses our prototype implementation of ComPoScan in detail. The results of our real-world deployment are presented in Section 6. Finally, Section 7 provides a discussion and Section 8 concludes the paper and provides directions for future work.

2. RELATED WORK

As mentioned earlier, existing 802.11 positioning systems (e.g., [2, 6]) have not considered the problem of concurrent communication and positioning. As a central part of the ComPoScan system we apply movement detection to deal with this problem.

The first, and as far as we know the only, 802.11-based system that emphatically focuses on movement detection is the LOCADIO system [11]. In their paper, the authors propose an algorithm that exploits the fact that the variance of signal strength measurements increases if the mobile device is moved compared to if it is still. To smooth the high frequency of state transitions an HMM is applied. The results in the paper show that the system detects in 87 percent of all cases whether the mobile device is in motion or not. Compared to our approach, the authors do not compare their system to other movement detection algorithms. Furthermore, the results are only based on emulation which means that the signal strength data is collected in a first step and then, later on, analyzed and processed to detect movement. This is a valid approach, but some real-world effects might be missed. Another fact that the authors of the aforementioned paper do not look into is the impact of periodic scanning to the communication capabilities of mobile devices. They just assume that a 802.11 network card is solely used for movement detection. Finally, all results are based on one single client, which means that variations in signal strength measurements caused by different wireless network cards are not taken into account.

Two GSM-based systems have also been proposed by Sohn et al. [16] and Anderson et al. [1]. The system by Sohn et al. is based on several features including variation in Euclidean distance, signal strength variance and correlation of strength ranking of cell towers. The system classifies data into the three states of still, walking and driving. By emulation on collected data they achieve an overall accuracy of 85 percent. The system by Anderson et al. detects the same states, but uses the features of signal strength fluctuation and number of neighboring cells. Using these features, they achieve a comparable overall accuracy to the former system. As for LOCADIO the results for both systems are only based on emulation, they also do not consider communication and the results are based on one client.

3. COMPOS CAN SYSTEM

For our system we assume that the mobile device that should be ComPoScan-enabled contains a 802.11 network card. This card should be able to perform active scans and monitor sniffs on a high rate (e.g., every 600 milliseconds). Further, the card should not include buffered results from a previous scan into the current scan result. For the area where ComPoScan should be deployed we assume that at least one access point is recognizable at all times by monitor sniffing and active scanning.

Our main goal for ComPoScan is to minimize the impact of scanning on concurrent communications. For this, we want to build a movement detection system that, based on signal strength measurements provided by monitor sniffing or active scanning, detects correctly whether the user is standing still or moving. If this is possible only active scans are required in case that the user is roaming around. However, we expect that it might be impossible to build a completely perfect movement detection system with 802.11. So this brings up a sub-goal: The movement detection system should be configurable in such a way that the user can define the kind of the error the movement detection system is producing. In case that the user is more interested in precise position estimates than in uninterrupted communications this scenario should be configurable. The other way around should also be supported.
ComPoScan works as illustrated in Figure 2. At startup, active scans are performed to collect signal strength values from as many access points as possible. Based on this data, the current state is calculated. If the system detects movement, it performs another active scan. In case that the system draws the conclusion that the user is standing still it switches to monitor sniffing for signal strength measurement. Based on this data, the current state is reevaluated and the system starts over again.

![Figure 2: The ComPoScan system.](image)

4. MOBILITY DETECTION

A central part of the ComPoScan system is movement detection based on signal strength. This section describes our experimental setup, gives an analysis of features used for movement detection, presents the used method and discusses our emulation results.

4.1 Experimental Setup

For our experimental setup, we describe the used hardware and software setup, the test environments and the details of the data collection process.

4.1.1 Hardware and Software Setup

To collect the signal strength measurements, we used an IBM Thinkpad R51 laptop running Linux kernel 2.6.22.12 and Wireless Tools 29pre22. To show that our approach works independent of a particular card, we use different network cards. For this, three network cards were chosen that are all quite frequently used today. We selected a Lucent Orinoco Silver PCMCIA card, a TRENDnet TEW-501PC PCMCIA card, and an Intel Centrino 2200 mini-pci card. The Lucent Orinoco card is a 802.11b only card. The TRENDnet card is based on the widely used Atheros AR5006XS chip-set and supports 802.11b, 802.11g, and 802.11a. Only 802.11b and 802.11g are supported by the Intel Centrino chip. However, all three network cards can be used for our purposes, because they all support monitor sniffing and active scanning.

For the Intel Centrino 2200 card, we used the ipw2200 driver in version 1.2.0^3. In the default settings, the driver caches a scan result for 3.45 seconds which means that an access point, that has been seen during the last 3.45 seconds, will appear in a subsequent scan result and even that it might be out of communication range. We modified the driver to discard old scan results before a new scan is performed because this property harms our movement detection system.

The driver of the TRENDnetcard needed modifications, too. For this card, we used the madwifi driver version 0.9.3.3^4. In the default settings, the driver caches scan results in the same way as the ipw2200 driver. The difference here is that

---

^3 [http://www.nongnu.org/orinoco/](http://www.nongnu.org/orinoco/)

---
six are available in more than half of the measurements. In fact, 19 access points weakly cover small parts of the hall and hence are only available in less than ten percent of all measurements.

4.1.3 Data Collection

For the two test environments, we collected signal strength data with two network cards at the same time. One network card uses monitor sniffing, the other one active scanning. This allows us to directly compare signal strength measurements taken by monitor sniffing and active scanning, because they are collected at the same time in exactly the same scenario. The network cards perform an active scan or a monitor sniff every 600 milliseconds.

To be able to compare different network cards, we collected data for each environment with two different hardware configurations. The first configuration uses the Intel Centrino and the Lucent Orinoco network cards. The Centrino network card is performing monitor sniffing and the Orinoco card carries out active scans. For the second configuration, the Centrino card is configured to perform active scans and the TRENDnet card collects data by using monitor sniffing.

For each test environment and each hardware configuration, we collected four different movement scenarios. The four scenarios consist of two slow walking scenarios and two fast walking scenarios. Slow walking is defined as an average movement speed below 0.7 m/s. Fast walking is defined if the movement speed is above this threshold. One of the slow walking and one of the fast walking scenarios comprise two movement transitions and the other scenarios include nine. A movement transition is defined as start walking or stop walking. The percentage of time where the person remains still is varied between different scenarios. We selected the parameters for these scenarios in such a way that the parameter space containing movement speed, number of transitions, and percentage of time where the person stands still is masked. All the scenarios are listed and described in Table 1. This table also names typical representatives for these kind of movements. The representation was what we had in mind when defining the movement scenarios to collect data.

### Table 1: Description of the different scenarios used for data collection.

<table>
<thead>
<tr>
<th>Speed</th>
<th>Transitions</th>
<th>% being still</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow</td>
<td>2</td>
<td>90</td>
<td>Meeting attendant</td>
</tr>
<tr>
<td>Fast</td>
<td>2</td>
<td>40</td>
<td>Student working</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in a lab</td>
</tr>
<tr>
<td>Slow</td>
<td>9</td>
<td>40</td>
<td>Student during lunch break</td>
</tr>
<tr>
<td>Fast</td>
<td>9</td>
<td>90</td>
<td>Office worker</td>
</tr>
</tbody>
</table>

To be able to investigate the impact of different times of the day, we collected all possible options once during typical office hours and once during evenings. In total, we collected more than eleven hours of signal strength measurements.

We used the aforementioned Tracer application to trace the walks of the persons collecting data. For this, we stuck labels on the floor so that each corner and each dead end of the walking track was marked. Each time the person carrying out signal strength measurements reached one of these labels the Tracer application was notified of the arrival at this particular landmark by a push of a button. Based on the trace of button clicks, we calculated the average movement speed between two landmarks. To be able to recognize still periods, another button was pressed each time the person started moving again.

The data for the fingerprint database was also collected using the Tracer application. We applied a grid of reference points to the operation area which includes 225 points with a spacing of 1.5 meter (see the blue markers in Figure 3(a)). During fingerprinting, we collected 120 signal strength samples at each reference point, resulting in a total of 27,000 samples. For the data collection of the fingerprint database, we used the Orinoco network card.

4.2 Feature Analysis

Movement detection using signal strength can be based on several features calculated from a sliding window of signal strength measurements. Previous research has explored the features of Euclidean distance, signal strength variance and rank correlation. The Euclidean distance feature can be calculated in several forms. First, the gap form where the Euclidean distance is calculated between the first and the last signal strength measurement in a sliding window. Second, the average form where the Euclidean distance is calculated between each consecutive signal strength measurement and then average together. Compared to previous work, in this paper we assume that signal strength measurements are only available for one access point to support movement detection with monitor sniffing. This assumption means that the rank correlation feature cannot be used, because this feature requires measurements from several access points to be able to rank them in terms of signal strength. The Euclidean distance feature collapses to the absolute difference in signal strength for one access point. The goal of this section is to both analyze how such features behave under movement and whether this behavior can satisfy a number of reliability requirements which are listed in Table 2.

The data collected as described in Section 4.1 allows us to evaluate, by means of emulation, the mentioned features with respect to these requirements. In the following, we focus on the feature of signal strength variance which is the feature that we later, based on our emulation results, will choose for our validation of the system. For the analysis, Figure 4 to 6 plot the median signal strength variance for each of the 64 different options in our data set within the categories still, slow and fast movement. The feature values are calculated using a ten measurements long sliding window. They are calculated for three access points from the Hopper building and for three access points from the Benjamin building. These six access points were chosen out of all measured access points to be representative in order to increase the readability of the graphs. Each single data point of the graph therefore represents the median feature value for a specific access point in one of the 64 options. Within each category, the data points are distributed based on when they were collected for readability of the graphs.

### Table 2: Requirements.

1. Detect using both monitor sniffing and active scanning measurements.
2. Work with different network cards.
3. Work in different physical environments.
4. Work with access points with different physical replacement.
5. Work both at day and at night.
6. For people with different mobility patterns.

A central feature of the ComPoScan system is the use of light-weight monitor sniffing in addition to invasive active
scanning. Therefore it is important that the chosen feature works with signal strength measured using either monitor sniffing or active scanning. Our data analysis confirms that both with active scanning and monitor sniffing measurements the signal strength variance increases with movement. Another important consideration for the system is that it should work with different network cards. In Figure 4, data is plotted marked and grouped with respect to the collecting network card. From the figure it can be noticed that the median signal strength variance changes similarly across the categories for all three network cards.

To consider the impact of different environments, Figure 5 plots the data marked and regrouped depending on where they were collected. From the plot it can be noticed that in both environments the median signal strength variance increases with movement. However, the spread in values is higher in the Hopper building than in the Benjamin building. In order to further analyze why this is the case, Figure 6 plots the data marked and regrouped for the six access points. The access points one to three are from the Hopper building and four to six are from the Benjamin building. For each environment, the access points are ordered by how often they were measured and their median signal strength. From the graph it can be observed that the signal strength variance is higher for frequently measured access points with a high median signal strength. Therefore the difference between the two environments can be explained by different distributions of weak and strong access points. This means that signal strength variance does not only depend on speed, but also on the strength of measured access points.

The median signal strength variance has also been analyzed with respect to the different mobility scenarios and the time of collection, but the graphs for these have been omitted. For mobility scenarios, the median signal strength variance showed the same behavior across the categories and had the same spread. In respect to time the same change in signal strength variance could be noticed across categories between day and night data.

The above analysis was based on the median signal strength variance split into three categories. Below, we will only consider the two categories of still and moving which are our target categories. Above, the median signal strength variance was considered. In the following the complete distribution
of the signal strength variance is considered for the two categories. Figure 7 plots such distributions for still and moving for the three network cards. This plot shows that the still distributions for Orinoco and Centrino tops at zero and TRENDNet at one. The moving distributions tops for Orinoco at ten, Centrino at eighth, and TRENDNet at 14. However, the overall shapes of the distributions are the same but with the TRENDNet distributions tending to include larger values.

In respect to the environment, Figure 8 plots the still and moving distributions for the two environments. The still distributions both top at zero, but the still distribution of the Benjamin building has a higher percentage of larger values than the Hopper distribution. The moving distributions both top at ten. The small differences in the distributions can again be attributed to the presence of different access points.

To summarize, from this analysis we can make several conclusions with regard to the listed requirements. First, signal strength variance is consistent when calculated from signal strength values measured using either monitor sniffing or active scanning. Second, signal strength variance calculated from measurements collected with different network cards share the same difference with respect to being still or moving. Third, with respect to different physical environments a minor variation was observed and further analysis identified this difference to be attributed to the physical replacement of access points in the areas. It was identified that strong access points show the largest signal strength variance in the different categories. Finally, no significant differences were identified at different times of a day and for different mobility scenarios.

### 4.3 Methods

Several detectors have been applied by earlier work to detect movement using the aforementioned features. Using our data, we have evaluated several of these and finally selected a Hidden-Markov Model (HMM) following Krumm et al. [11] as the best option. In addition to the HMM detector we evaluated a simple Naive Bayes detector and two AdaBoost detectors instantiated with ZeroR and Naive Bayes as basic detectors. In the next section, we will provide some emulation results that will support the choice of the HMM. The primary difference between the HMM detector and the other three detectors is that HMM is able to take previous feature values into account and can thereby minimize that the detector is immediately flipping back and forth between detecting movement and detecting still. A drawback with the HMM method is that it is only able to work with one feature type whereas the others can use several.

The used HMM has two states: still and moving. Probabilities are assigned to each state for staying or transition to the other state. The probability of being in either states are initially set to be equal. Each state also has a distribution associated with it that gives the probability of observing a feature value in this state. In each prediction step, a set of consecutive feature values within a sliding window is used by a Viterbi algorithm to calculate the most likely sequence of state changes in the model. The estimated state then is the ending state of the calculated sequence. The distributions are calculated from a set of training data. In this work, the Gaussian kernel method is applied, because it creates more generalizable distributions than the histogram method. Based on initial experiments, a standard deviation of 0.1 was selected for generating the still distributions and a standard deviation of 1.5 for generating the movement distributions by the kernel method. An important point is that the feature analysis showed that such distributions can be used for different network cards and for environments with a different access point availability. This means that the system does not need to be trained for each specific deployment which will be further supported by our emulation and validation results.

### 4.4 Emulation Results

The purpose of this emulation is to identify a good detector for the ComPoScan system and to find the parameters for the identified detector. These should be used in our validation of the system. An important goal for a detector is that it is good at detecting movement, but also that it allows us to make different trade-offs to either favor communication or positioning.

Several types of emulations have been run to evaluate the detectors on the collected data. Emulations for Naive Bayes using all features, the HMM with gap Euclidean distance, the HMM with average Euclidean distance, and the HMM with signal strength variance have all been run in an extension to the Loceva toolkit [7] implemented by the authors. For the HMM implementation we used the Jahmm library\(^4\). The emulations for AdaBoost using Decision Stumps and AdaBoost using Naive Bayes were run in the Weka toolkit [18]. Initially, emulations were run to find good values for the window size used in the feature calculations and for the history size used by the HMM detectors. The results were that higher values of window size and history size made the detectors better at detecting still but worse at detecting movement. For our data, this will improve a detector’s overall accuracy because our data contains more still than moving data. However, for the ComPoScan system the overall accuracy is less important than a detector’s ability to detect movement. Therefore, we focused on finding a window size and a history size that would make the detectors good at detecting movement without sacrificing too much on the overall accuracy. Another reason for keeping the window size low is that this minimizes the startup time before the system can start making predictions and for the history size it minimizes the computational requirements for the Viterbi algorithm used by HMMs. Based on these criteria, we selected a window size of ten and also a history size of ten.

To compare the detectors eightfold cross-validation was applied where data was split into folds depending on the scenario collection round. This makes sure that test and training data have not been collected at the same point in time. For the emulation output, movement was chosen as a positive output and still as a negative output. This means that

\[^4\]http://www.run.montefiore.ulg.ac.be/~francois/software/jahmm/
we can count the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). From these counts we can calculate the true positive rate \( tp = 100\% \times TP/(TP + FN) \) and the false positive rate \( fp = 100\% \times FP/(FP + TN) \). By varying the parameters of the HMMs and setting different selection thresholds for Naive Bayes and the two AdaBoost methods a curve of pairs of \( tp \) and \( fp \) can be plotted. Such a graph is known as a Receiver Operating Characteristic (ROC) curve [18]. The curve shows which different trade-offs can be made in terms of \( tp \) and \( fp \). The more a graph of a detector stretches towards the upper left corner the better the detector performs.

The ROC curves for Naive Bayes, the two AdaBoost detectors, and HMM with signal strength variance are shown in Figure 9. The curves show that there are only small differences among the different detectors. The AdaBoost detector performs best when the \( tp \) is below 60, while the HMM with signal strength variance performs best above 80. The Naive Bayes detector performs worst regardless of the interval. For ComPoScan we are interested in a detector that is good at detecting movement and therefore we are interested in a detector that can maximize the \( tp \) without increasing the \( fp \) too much. Therefore, the HMM with signal strength variance is the best choice. We also have emulated the HMM with the other features and the results are shown, focused on the \( tp \) and \( fp \) intervals of interest, in Figure 10. Again, there are only small differences between the detectors. The HMM with gap Euclidean distance performs best with \( tp \) below 60, but in the interval above 80, which we are interested in, the HMM with signal strength variance performs best closely followed by the HMM with average Euclidean distance.

To validate our system, we need to fix the parameters for the prototype. For the HMM with signal strength variance the parameters to fix are the two transition probabilities of the HMM. To solve this, in Figure 10 two lines are plotted which marks the optimal performance when treating the value of errors in different ratios. The red line treats false negatives and false positives equally, and the blue line treats them in the ratio one false negative to three false positives. On each of the lines we choose one set of parameters as illustrated in Figure 10.

![Figure 9: Emulation results as ROC.](image)

For the parameter set with a probability of 0.011 to change from moving to being still and 0.0011 to change from still to moving, we ran further emulations in order to evaluate how the chosen detector addresses the design requirements listed in Table 2. The emulations were run as cross validation with a different number of folds depending on how many categories they should be split in. Table 3 to Table 5 list some of the emulation results where each table entry is named by the test data. So, the first entry in Table 3 is from an emulation with Centrino as test data and from the other two cards as training data and so on.

The results for monitor sniffing and active scanning showed that the detector is working equally well using both monitor sniffing and active scanning measurements. The results in Table 3 highlight that there are some variation across network cards. The TRENDnet network card is best at detecting movement whereas the Centrino card is worst. If we compare these results with the distributions for the different network cards shown in Figure 7, we notice that the TRENDnet distribution for moving is right shifted compared to the other distributions. This means that when testing with TRENDnet on training data from the two other network cards the probability of the detector predicting still is decreased. So, there are some variations across network cards, but it is mainly changing the detector’s trade-off between predicting still or moving, not making the detector unable to detect movement at all.

The results in Table 4 show that in the Benjamin building the detector was worse at detecting still compared to the Hopper building. On the other hand, in the Benjamin building movement was better detected. Comparing the distributions for the two environments in Figure 8, it can be noticed that for the Benjamin building the still and moving distributions do have a larger overlap than for the Hopper building. This can both be attributed to the absence of walls lowering the variations in signal strength but it can also be attributed to the difference in access points availability. To analyze this claim, Table 5 provides results split over the different access points. For the weakest and least measured access points one and four, we obtain the best still detection rate because of the lower signal strength variance as identified from Figure 6. The results also indicate that the access points one to three from the Hopper building give better still detection results than the access points four to six in the Benjamin building.

In all the previous results the feature values were calculated from measurements for a single access point. However, you can easily extend the calculation of the feature values to use measurements from several access points. With active scanning, multiple access points will normally be measured, but
sometimes also monitor sniffing measurement will be made to several access points that are on the same channel or on close channels. For signal strength variance, we extend the calculation to multiple access points by calculating the average value over the signal strength variance calculated for each single access point. Denoting the number of access points as \( k \), we ran emulations with a different size of \( k \). The results in Table 6 indicate that both the detection of moving and still improves when increasing \( k \). In our validation we therefore use a detector that uses the highest \( k \) possible given the current measurements.

<table>
<thead>
<tr>
<th>Card</th>
<th>True positives (%)</th>
<th>False positives (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centrino</td>
<td>76.2</td>
<td>12.6</td>
</tr>
<tr>
<td>Orinoco</td>
<td>89.3</td>
<td>35.8</td>
</tr>
<tr>
<td>TRENDnet</td>
<td>93.3</td>
<td>25.4</td>
</tr>
</tbody>
</table>

Table 3: Network cards.

<table>
<thead>
<tr>
<th>Environment</th>
<th>True positives (%)</th>
<th>False positives (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hopper Building</td>
<td>83.6</td>
<td>20.4</td>
</tr>
<tr>
<td>Benjamin Building</td>
<td>89.9</td>
<td>41.7</td>
</tr>
</tbody>
</table>

Table 4: Physical environments.

<table>
<thead>
<tr>
<th>AP</th>
<th>True positives (%)</th>
<th>False positives (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>78.4</td>
<td>10.7</td>
</tr>
<tr>
<td>2</td>
<td>97.8</td>
<td>43.4</td>
</tr>
<tr>
<td>3</td>
<td>97.1</td>
<td>39.3</td>
</tr>
<tr>
<td>4</td>
<td>95.7</td>
<td>32.8</td>
</tr>
<tr>
<td>5</td>
<td>90.4</td>
<td>50.9</td>
</tr>
<tr>
<td>6</td>
<td>98.6</td>
<td>63.2</td>
</tr>
</tbody>
</table>

Table 5: Access points (AP).

We compare the emulation results achieved by ComPoScan with the results obtained by traditional positioning. The average positioning error with traditional positioning was 3.81 meters and with ComPoScan it was 3.74 meters. The results indicate that ComPoScan on average actually improves the position accuracy with two percent. However, for two of the eight scenarios ComPoScan decreases the accuracy. From the emulation we can therefore conclude that ComPoScan does not have a major impact on the position accuracy and might even improve it in some cases.

5. PROTOTYPE IMPLEMENTATION

We wanted to see if our ComPoScan system works in real-world deployments in the same way as the emulation results suggest. To be able to deploy the system in real-world environments, we implemented a prototype that runs on Linux and supports any 802.11 network card that is able to perform monitor sniffs and active scans.

Based on the emulation results, we selected only the HMM signal strength variance detector to be implemented. Further, we kept the parameters of this detector easily configurable to make sure that, during our validation, we can switch between different configurations to trade communication capabilities against positioning accuracy.

The implementation is structured into three different parts that deal with diverse tasks:

- **Driver:** We wanted the prototype to support at least Atheros-based wireless network cards. The reason for this is that we owned a NETGEAR WG511T network card that contains the Atheros AR5212 chip-set that has not been used for data collection in the emulation. The driver we used for this card is modified in the same way as described in Section 4.1.1.

- **Signal strength measurement system:** Depending on the results calculated by the movement detection system, an active scan or a monitor sniff is performed to collect signal strength measurements. In case that the movement detection system cannot calculate any result (e.g., a lack of sufficient signal strength samples) an active scan is executed. To invoke an active scan or a monitor sniff, we used the Loclib library [7]. This library collects signal strength measurements from the kernel driver and makes them available to user-space applications.

- **Movement detection system:** After the collection of a signal strength measurement, the data is stored with other recent measurements in a ten-entries sliding window. For all access points that are available in at least eight of the measurements, the signal strength variance is calculated. These values are then forwarded to the HMM to decide whether the mobile device that provided the signal strength measurements is currently moving or not. For the HMM implementation we used the same
Furthermore, to be able to evaluate the impact of ComPoScan on communication capabilities and the positioning accuracy of a mobile device, we additionally implemented three sub-systems. The so-called network measurement system gauges throughput, delay and packet loss. For this, we utilized a tool called iperf. Iperf is a client-server application that measures the maximum throughput achievable over a given link. To measure round trip delay and packet loss we implemented a client application that sends out ping requests every 100 ms. The corresponding server application sends back a ping response every time a ping request packet arrives. The identification number, contained in each ping request packet, is copied into the response. In this way, the client is so able to calculate the round trip delay by subtracting the time when a ping request packet carrying a certain identification number was sent out from the time when the corresponding ping response packet arrived. In case that no response arrives, the request or response packet must be lost during transmission.

The second and third sub-systems are required to calculate position estimates based on signal strength measurements. For this, we implemented a signal strength normalization method that makes signal strength data taken from different network cards directly comparable. The method we selected is published by Kjærgaard [9]. It finds a linear function to match signal strength measurements of two different cards by using the least squares analysis. This approach works pretty well and it can easily be applied, because only at a few different locations signal strength measurements from the cards that should be normalized are required. However, normalization can be completely avoided by using the technique of hyperbolic location fingerprinting proposed by Kjærgaard et al. [10]. On top of this stands the positioning system. As positioning algorithm we selected one of the probably most studied ones: The Gaussian fit probabilistic algorithm proposed in [6].

![Figure 11: Architecture of the prototype implementation.](image)

In Figure 11 the architecture of the ComPoScan prototype implementation is illustrated. The sub-systems belonging to the ComPoScan system are marked by gray boxes. The three additional sub-systems required to evaluate the prototype system are depicted in white boxes. The arrows show how information is distributed between different parts of the system.

For the evaluation of the prototype we also developed a small tool that writes a timestamp to a trace file each time its button is pressed. The person who validates the prototype is supposed to press this button each time walking is started or stopped.

6. REAL-WORLD VALIDATION

In this section, we present our results obtained from prototype deployment during a period of more than one week. The system is deployed in eight buildings of the University of Aarhus, Denmark (see Figure 12). The movement detection accuracy is illustrated from walks through the Ada, Babbage, Benjamin, Bush, Hopper, Shannon, Stibitz, and Turing buildings. As the positioning system is only available in the Hopper building, the position accuracy measurements are only collected from this place. The network conditions are also investigated showing the benefit of the system’s configurability by using two parameter sets.

![Figure 12: This map shows the names of the buildings where the ComPoScan system has been validated in real-world deployments.](image)

For most real-world experiments we configured the HMM detector in such a way that we favored stable network conditions over position accuracy. Or in other words, we wanted to make sure that ComPoScan only performs active scans if it is quite sure that the person in question is moving. If it is not stated otherwise, we used the following parameters: The probability to change from moving to being still is 0.0011, the probability to change from being still to moving is 0.011, the window size is ten and the history size is ten.

6.1 Movement Detection Accuracy

To study the movement detection accuracy of our ComPoScan system, we recorded a typical route a member of the University of Aarhus would go from an office of the Hopper building to the Cafeteria located in the Benjamin building to pick up a cup of coffee. On the way back the walk contains stops at different locations to chat with colleagues and to pick up mail and printouts. The path leads additionally through the Ada, Babbage, Bush and Turing buildings.

We recorded the walk once in the morning when many people move around in the buildings and once during the night when the building is abandoned. The walk lasts for more than 25 minutes and contains eleven stops. In both walks, the person who walked around is standing still in 79 percent of the time.

The real movement of the two walks is indicated in Figure 13(a). Movement as perceived by ComPoScan is depicted in Figure 13(b). As we see from the two figures, still periods are quite often detected for both walks. The walk performed in the morning shows correctly detected still periods in 87 percent of all real still periods. This is in contrast to 94 percent for the walk during the night. Furthermore, movement is also pretty often detected. However, the correct detection rates are
a bit lower here. For the walk at night ComPoScan achieves 54 percent and 65 percent for the walk in the morning. The main reason why ComPoScan detects moving periods better during the day is the following: During the time we performed our walk many people arrived at the office to start working or walked to other offices or meeting rooms to attend meetings. Moving people increase the signal strength variance, because each time they walk into the path a radio signal is traveling they attenuate the signal. This means that the signal strength measurements are a bit more broadened. The scattered measurements add to the signal strength variance caused by movement which means that movement is easier to detect.

A further analysis of the data shows that there is always a delay between real movement and movement reported by the HMM detector. Three reasons cause this delay: First, if we assume that ComPoScan detected the motionless state correctly then only monitor sniffs are performed. During a monitor sniff only beacons emitted from access points are examined. These results are stored for 600 milliseconds before forwarded to the HMM detector. This procedure is chosen to be similar to an active scan. Even though there might be room for improvements, our further analysis shows that this delay accounts only slightly to the overall delay. The second reason is the sliding window we utilize to calculate the signal strength variance. To smooth variations in the signal strength measurements, we applied a window size of ten entries. It takes some measurements to propagate an increased signal strength variance through this window. Third, the history required by the Viterbi algorithm to be able to calculate the most likely sequence of state changes also adds up to the overall delay. For instance, the walk during the night shows an average detection delay of 8.6 seconds with a standard deviation of 13.3 seconds. The minimum delay we observed is only 156 milliseconds and the maximum detection delay is 13.4 seconds. The morning walk shows similar delays.

We also configured ComPoScan to use the second parameter set as defined in Section 4.4. Compared to the previously selected parameter set, this set differently trades movement detection against still period detection. We expect to see the percentage of correctly detected movements to go up and the rate of correctly detected still periods to go down. The comparison of the two parameter sets is based on data we have collected in the Shannon building. This walk contains five stop periods which represent about 67 percent of the more than 13 minutes walk. The results are listed in Table 7.

Table 7: Movement detection accuracy for different configurations in the Shannon building.

<table>
<thead>
<tr>
<th></th>
<th>(0.011, 0.0011)</th>
<th>(0.5, 0.00011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True positives (%)</td>
<td>51.25</td>
<td>78.13</td>
</tr>
<tr>
<td>True negatives (%)</td>
<td>99.23</td>
<td>74.86</td>
</tr>
<tr>
<td>Overall accuracy (%)</td>
<td>82.42</td>
<td>76.05</td>
</tr>
</tbody>
</table>

Even though the overall accuracy drops by six percent, we see that the second configuration works like expected. The true positives rate increases from 51.25 to 78.13 percent, which means that the second configuration correctly detects movement better than the first configuration. However, this improvement comes with the cost of a drop of the true negatives rate. The second configuration only recognizes still periods in 75 percent correctly. Knowing this it is clear that the overall accuracy is a weak measure for our system. The overall accuracy depends on the movement pattern and the selected configuration of our system. To make results comparable the true positives rate and true negatives rate should be used.

Additionally, the results presented in Table 7 also show that ComPoScan works in different buildings without any further training. For all our validation walks we only used the data collected in the hall of the Benjamin building to train the HMM detector. To back up our claim, that ComPoScan does not require any local training to be functional, we went to the Stibitz building for another round of validation. The data reveals that movement is correctly detected in 67.07 percent of all walking periods and still times are correctly detected in 99.23 percent of all cases. This is consistent with the other results we have presented.

In summary, we have shown that ComPoScan is able to work well in different buildings, with different configurations, during different times of a day, and with a new network card.

6.2 Positioning Accuracy

As already mentioned, only on the second floor of the Hopper building a 802.11-based positioning system was deployed. So, all results presented in this section are collected in this part of the building. We define position error as the Euclidean distance between the real position of the user and the position estimate computed by the positioning system. The term average position error refers to a set of position errors averaged over time.

The positioning system is set up in the same way as described in Section 4.1.3. The parameters for the normalization system are gathered by collecting signal strength samples with both the Orinoco and NETGEAR network card at five randomly selected positions within the operation area. Based on the least square analysis the parameters are calculated.

During validation, each time the signal strength measurement system invokes an active scan the measurements are copied and forwarded to the signal strength normalization system. After normalizing the measurements, the values are send to the positioning system. The positioning system uses only measurements obtained from one active scan to calculate a position estimate. Before a new position estimate is calculated the positioning system is reset to discard any knowledge learned from a previous measurement. We are aware of the fact that tracking technologies might improve the positioning accuracy, but we wanted to keep the positioning system as simple as possible to get a clear insight into the impact of ComPoScan on the positioning accuracy.
Signal strength measurements collected from a monitor sniff are omitted, because they usually contain only a sub-set of all available access points. While it might be possible to increase the positioning accuracy by using monitor sniffing results, we want to investigate how the position accuracy drops if movement is not correctly detected and hence signal strength measurements from active scans are missing.

The Tracer application described in Section 4.1.3 was used to collect real position information of the user carrying out the real-world validation. For this, we stuck labels on the floor to mark prominent places of the path the user was supposed to walk. Based on the Tracer’s trace and known positions of the labels, we are able to calculate the real position of the user during the walk. The real position information is later on compared to the position estimates computed by the positioning system.

We selected two walks of 370 seconds each from our data to investigate the positioning accuracy in detail. During one walk ComPoScan was activated, while during the other walk only the positioning system was running. In the latter setup, the positioning system calculates position estimates every 600 milliseconds. We refer to this setup as traditional positioning. Real movement and movement detected by ComPoScan are presented in Figure 14(a) and 14(b), respectively. The real movement for the two walks slightly differs, because the person moving around paused a bit more during the ComPoScan walk.

Figure 14(c) depicts the positioning error for traditional positioning as well as the ComPoScan system. On average, ComPoScan achieves an error of 4.68 meters whereas traditional positioning is slightly worse by obtaining 4.74 meters. From the graph we see that the curves generated by the two systems look quite similar. During still periods we observe that the positioning error produced by ComPoScan is more stable than what can be achieved by traditional positioning. This is consistent with the average positioning error only calculated for real still periods. ComPoScan achieves 4.06 meters in contrast to 4.16 meters produced by traditional positioning. On the other side, we see from the figure that the ComPoScan position accuracy is worse than what can be achieved by traditional positioning during periods of real movement. On average, the positioning error is 8.22 and 8.12 meters, respectively. The reason for this is that ComPoScan calculates only a new position estimate if movement is detected. From the previous section we know that ComPoScan detects movement always a bit later than it actually happens. This means if a person already started walking, ComPoScan is still perceiving the person to be motionless for a short period of time. The increase in the positioning error is caused by this delay.

All our traditional positioning data shows an average positioning error of 4.68 meters. In contrast, the average positioning error over all ComPoScan data is 4.44 meters. If we compare this to the emulation results presented in Section 4.4, we see a difference of 0.87 and 0.70 meters, respectively. The difference is caused by using different network cards to collect the fingerprint database and while performing signal strength measurement for positioning. Additionally, the signal strength normalization system also contributes to the positioning error as shown by Kjærgaard [9].

We also configured ComPoScan to use the second parameter set we have chosen in the emulation section. This parameter set sets the state change probability of the HMM detector to switch from moving to being still to 0.00011 and the vice versa probability to 0.5. These parameters are supposed to be more positioning friendly and hence we expect the positioning error goes down. To be able to compare the positioning performance with these parameters to the previously used ones, we walked the path a third time. Figure 15 shows the position error for the two configurations. The curve for the previously used parameter set is exactly what we have seen in Figure 14(c). If we compare this curve with the curve produced by the second parameter set, we see that the previously used parameters are outperformed by the newly applied parameters. On average, the newly applied parameters achieve a positioning error of 3.68 meters. This is exactly one meter less than what we obtained from the previously used parameter set.

To sum up, we have shown that ComPoScan does not harm the positioning accuracy at all. We even showed that the positioning error goes up a bit during movement periods and that it slightly drops during still periods. Further, ComPoScan’s configuration parameters can be used to define its sensibility to compute position updates.

### 6.3 Communication Capabilities

One of the reasons why we came up with the idea of ComPoScan is that communication is quite weak in terms of throughput, delay, and packet loss if the 802.11 network card is used for positioning at the same time. ComPoScan is build in such a way that it trades communication capabilities against position accuracy. For this it switches to light-
weight monitor sniffing when it detects that the person carrying a ComPoScan-enabled mobile device is standing still. During monitor sniffs, uninterrupted communication with the access point the mobile device is associated with is possible. In case ComPoScan detects movement it switches to invasive active scanning to collect enough data to enable the positioning system to calculate an accurate position estimate.

The network measurement system as described in Section 5 is utilized to generate the results presented in this section. We used an extra Apple Airport Extreme access point directly connected to a Fujitsu-Siemens Lifebook T4010 laptop running iperf and the ping server. This setup guaranteed that the only bottleneck is the wireless channel and not the wired network.

We configured iperf to send a UDP stream for six minutes with a bandwidth of 17.6 MBit/s. This value was determined by a stepwise increase until no gain could be achieved. The simple UDP transport protocol is selected because the more sophisticated protocols often bring congestion avoidance strategies that may interfere with our measurements. As we are the only participant in this network, the UDP measurements are the upper limit of what is achievable. We measured the throughput in the smallest time interval supported by iperf: Every 0.5 seconds. During the time iperf was sending data, we measured the round trip delay by using our self-developed application. We configured it to send a ping request every 100 milliseconds.

For the throughput, delay, and packet loss measurements, we walked around in the Hopper building for six minutes. During this walk, we stopped five times representing 86 percent of the total time. To be able to compare our ComPoScan system, we repeated the walk while performing active scans every 600 milliseconds as a traditional positioning system would request to do.

In Figure 16(a) the real movement for both walks is presented. Figure 16(b) shows how ComPoScan perceives still and movement periods. The correctly detected rate for movement here is 58 percent and 84 percent for being still. The throughput results are depicted in Figure 16(c). ComPoScan is able to transfer 638.3 MBytes during the six minutes of the experiment whereas a traditional positioning system reduces the amount of data being transferred to 5.2 MBytes. This corresponds to an improvement of factor 122. On average, a throughput of 12.8 MBit/s and 0.1 MBit/s, and a standard deviation of 5.9 MBit/s and 0.07 MBit/s is achievable, respectively. During detected still periods, ComPoScan is able to transfer 14.9 MBit/s on average and 2.9 MBit/s during detected moving periods. The reason why ComPoScan achieves such a high number during detected movements is the detector update delay. A detector update is available every 600 milliseconds right after a monitor sniff or an active scan returns signal strength measurements.

If we compare the round trip delay, we see a similar picture (see Figure 16(d)). For ComPoScan, on average, the delay goes down to 46.97 milliseconds during detected still periods and up to a maximum of 1054 milliseconds during detected moving periods. The average delay during movement is 186.36 milliseconds. This sums up to a total average delay of 53.46 milliseconds. In contrast, traditional positioning achieves only an average round trip delay of 566.58 milliseconds and peaks around 3193 milliseconds.

The packet losses are related to the delay. During active scans no data can be transmitted meaning that the data is dropped after the different buffers provided by the network stack of the kernel and the network card driver are filled up. Figure 16(e) shows the packet loss for ComPoScan and traditional positioning during the walks. For ComPoScan, the packet loss spikes each time movement is detected (e.g., around 70, 100, and 200 seconds). During these periods ComPoScan drops around 72.79 percent of the packets. Only 3.49 percent of the packets are dropped during detected still periods. On average, ComPoScan drops 16.07 percent of all ping request and ping response packets. This is in contrast to traditional positioning where 89.78 percent of the packets are dropped. The reason why we see more than three percent of packets being dropped by ComPoScan during still periods is how we count packet loss. We consider the state ComPoScan reports during the generation of the ping request packet. So, for example, it might happen that a ping request is sent out when ComPoScan detects a still period. Directly after the packet left the mobile device, ComPoScan recognizes movement and starts an active scan. So, the ping response packet generated by the server application is then dropped, because it can not be delivered.

![Figure 16: Communication capabilities.](image-url)
accuracy. The downside of this is that the communication capabilities might be affected. In the following, we investigate how severe the impact is.

Figure 17 shows the throughput for both configurations. The spikes in throughput are not always at the same positions in the graph, because the two different configurations influence the movement detection system of ComPoScan inducing different detection results. Further, the person who validated the system walked with slightly different speeds and stayed still a bit longer at the different places during the two walks. However, as the walks are both equally long in terms of time and contain the same number of still periods and still times, the impact of the two configurations can be compared by looking at the average values over the total walking time.

Figure 17: Throughput for the two different configurations of ComPoScan.

For the second configuration, the average throughput is 10.01 MBit/s with a standard deviation of 10.6 MBit/s. This on average is 2.2 MBit/s less than what we achieved in the previous validation experiment. The total throughput during the six minutes of the experiment for the second parameter set is only 482.8 MBytes which is more than 155 MBytes less compared to the first parameter set. As shown by the graph, during real still periods ComPoScan using the second configuration switches quite often back and forth. This is the reason why we see such a huge standard deviation. The average delay draws a similar picture: On average, the round trip delay is 81.98 milliseconds compared to 53.46 milliseconds drawn from the other configuration. The maximum delay for the second configuration is on the same level as what we observed for the first configuration. For packet loss, the second parameter set produces a loss rate of 49.62 percent on average. In comparison to the results obtained from the first parameter set this is an increase of 33.55 percent.

To conclude this section, we have shown that ComPoScan generates communication conditions that can be used for meaningful data transfer. Further, we investigated two configurations to show that ComPoScan can be configured to favor communication capabilities.

7. DISCUSSION

In the literature (e.g., [4, 15]) it is reported that performing active scans regularly consumes more battery power than not scanning at all. For instance, Brunato et al. [4] state that their HP iPAQ H5450 PDA having the 802.11 network card switched off lives for 228 minutes. If the network card is switched on and associated with an access point without the network card switched off, scanning actively, and performing monitor sniffing the PDA’s battery is depleted in 140 minutes. Inspired by these results, we investigated the battery lifetime of our IBM R51 laptop while having the network card switched off, scanning actively, and performing monitor sniffing. We followed the recommendations for battery lifetime measurements listed in [3]. Our experiments showed that the six cell Li-ion battery of our laptop provided energy for 192 minutes if the network card was switched off. In case that active scans were performed every 0.6 seconds, the laptop lived for 177 minutes. The lifetime is increased to 184 minutes if monitor sniffing is performed. These results show that switching on the network card and selecting different scanning methods matters in terms of battery lifetime. However, the impact is smaller compared to the results shown by Brunato. The reason for this is that the battery drain caused by the different hardware components (e.g., CPU, memory, graphic card) of a PDA is lower compared to a laptop. The impact of the network card is smaller for a laptop than for a PDA. However, for resource-constrained devices such as a PDA, ComPoScan might also be able to increase their lifetime.

For our movement detection system we heavily rely on monitor sniffing. To be able to perform a monitor sniff, the 802.11 network card is configured to work in monitor mode. This mode sets the network card into a listening state that allows to receive frames sent from a wireless network the network card is not associated with. Even better, frames from channels close to the channel the network card is using for communication can be retrieved. Nowadays, most available network cards support monitor mode. For instance the Intel Centrino chip-sets (e.g., 2100, 2200, 2915, 3945) and nearly all Atheros chip-sets (e.g., AR5002G, AR5004X, AR5005, AR5211, AR5212) as well as older chip-sets such as the Lucent Orinoco chip-sets support monitor mode. Unfortunately, MS Windows does not support monitor mode by the NDIS driver interface which is why many drivers for this operating system do not support monitor mode. On the other hand, Linux and most BSD derivatives provide a wide range of drivers with build-in support for monitor mode. If the demand for monitor mode grows, we can expect to see more drivers for MS Windows supporting monitor mode as well. Therefore, enabling ComPoScan on MS Windows is just an implementation issue.

An easy and simple way to mitigate the impact of scanning on communications is to reduce the scan frequency. However, this solution comes with the drawback that the positioning error increases dramatically. For instance, let’s imagine that an active scan is performed only every four seconds. This means that the positioning system is only updated every four seconds with new signal strength measurements. In four seconds a person can walk up to six meters if we assume a descent walking speed of 1.5 m/s. So, in this scenario, on average, three meters have to be added to the positioning error of the positioning system. In indoor environments three meters matter, because they distinguish between different rooms. From our point of view, such an approach is not a solution.

Nowadays, modern hard-disks as part of laptops often contain accelerometers to protect the drive in case it is dropped accidentally. Many cell phones also contain such accelerometers to detect automatically if a picture is taken in landscape or portrait orientation. So, the question is if these accelerometers matter, because they distinguish between different rooms. From our point of view, such an approach is not a solution.

An easy and simple way to mitigate the impact of scanning on communications is to reduce the scan frequency. However, this solution comes with the drawback that the positioning error increases dramatically. For instance, let’s imagine that an active scan is performed only every four seconds. This means that the positioning system is only updated every four seconds with new signal strength measurements. In four seconds a person can walk up to six meters if we assume a descent walking speed of 1.5 m/s. So, in this scenario, on average, three meters have to be added to the positioning error of the positioning system. In indoor environments three meters matter, because they distinguish between different rooms. From our point of view, such an approach is not a solution.

Nowadays, modern hard-disks as part of laptops often contain accelerometers to protect the drive in case it is dropped accidentally. Many cell phones also contain such accelerometers to detect automatically if a picture is taken in landscape or portrait orientation. So, the question is if these accelerometers matter, because they distinguish between different rooms. From our point of view, such an approach is not a solution.
nipsent in mobile devices, our system might be extended to make use of them and further improve movement detection.

8. CONCLUSIONS

The primary contribution of this paper is the novel ComPoScan system that can mitigate the effect of scanning on concurrent communications. ComPoScan is based on movement detection to switch adaptively between lightweight monitor sniffing and invasive active scanning. Additionally, we provide an evaluation of the proposed system both by emulation and by validation in a real-world deployment. The emulation showed that our movement detection system works independently of the environment, the network card, the signal strength measurement technology, and number and placement of access points. We also showed that ComPoScan does not harm the positioning accuracy of the positioning system. By validation in a real-world deployment, we provided evidence for that the real system works as predicted by the emulation. In addition, we provide results for ComPoScan’s impact on communication where it increased throughput by a factor of 122, decreased the delay by a factor of ten, and decreased the percentage of dropped packages by 73 percent. Furthermore, as mentioned in the discussion, the system is also able to decrease the power consumption.

In our ongoing work we are trying to address several issues. These are: First, conceive a system that can make further use of monitor sniffing measurements for updating the position estimate without switching into active scanning. Second, switch between monitor sniffing and active scanning dependent on other metrics (e.g., network traffic, user preferences). Third, evaluate the impact of using accelerometers to implement movement detection. Fourth, implement our system on a smaller platform (e.g., PDA) that allows us to better evaluate the power savings of our system.

Acknowledgments

The authors acknowledge the financial support granted by the Deutschen Forschungsgemeinschaft (DFG), the European Science Foundation (ESF) and the software part of the ISIS Katrinebjerg competence centre.

9. REFERENCES