Improving vertical handover performance for RTP streams containing voice

Using network parameters to predict future network conditions in order to make a vertical handover decision

DANIEL YUNDA LOZANO

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2007

COS/CCS 2007-08

Improving vertical handover performance for RTP streams containing voice

Using network parameters to predict future network conditions in order to make a vertical handover decision

Daniel Yunda Lozano

dylozano@kth.se

February 28, 2007

In partial fulfillment of the requirements for the degree of Master of Science

Industry Supervisors: Ian Marsh and Martín Varela.

Swedish Institute of Computer Science (SICS)

Examiner: Prof. Gerald Q. Maguire Jr.

School of Information and Communication Technology (ICT)

Royal Institute of Technology (KTH), Stockholm, Sweden

Abstract

Wireless local area networks WLAN and Voice over IP technologies enable local low cost wireless telephony, while cellular networks offer wide-area coverage. The use of dual mode WLAN-cellular terminals should allow cost savings by automatically switching from GSM to WLAN networks whenever it is feasible. However, in order to allow user mobility during a call, a handover procedure for transferring a call between the WLAN interface and the cellular network should be defined. The decision algorithm that triggers such a handover is critical to maintain voice quality and uninterrupted communication. Information or measurements collected from the network may be used to anticipate when the connection will degrade to such a point that a handover is desirable in order to allow a sufficient time span for the handover's successful execution. It is the delay in detecting when to make a handover and the time to execute it that motivates the need for a prediction.

The goal of this thesis is therefore to present a method to predict when a handover should be made based upon network conditions. We selected a number of WLAN and VoIP software tools and adapted them to perform the measurements. These tools allowed us to measure parameters of the WLAN's physical and link layers. Packet losses and jitter measurements were used as well. We have assumed that there is ubiquitous cellular coverage so that we only have to be concerned with upward handovers (i.e, from the WLAN to the cellular network and not the reverse). Finally we have designed and evaluated a mechanism that triggers the handover based in these measurements.

Sammanfattning

WLAN, trådlöst lokalt nätverk, och IP-telefoni tillsammans gör det möjligt med billig trådlös telefoni, samtidigt som mobiltelefoninätverk erbjuder stor signal beläggning. Att använda WLAN-mobil med dubbla hårdvaruterminaler skulle ge en kostnadsreducering genom att automatisk byta från GSM till WLAN när det är möjligt. Emellertid för att kunna flytta pågående samtal mellan ett WLAN- och ett mobilt gränssnitt, måste en handovermekansim definieras. En beslutsalgoritm som utlöser sådan handover är av stor vikt för att bibehålla röstkvalitet och oavbruten kommunikation.

För att tillåta ett tillräckligt tidsspann för handoverns utförande kan information tagen från nätverket användas för att förutse när kommunikationen ska degraderas till en sådan punkt att en handover är önskvärd.

Förseningen i detekteringen när en handover ska ske och tiden för utförandet motiverar behovet av förutsägelse. Det här exjobbet introducerar en metod som förutsäger när handover ska börja baserade på nätverksförhållandena. Vi har valt några WLAN och VoIP-program och anpassat dem för att genomföra mätningarna. Programmen tillät oss att mäta WLANs parameter för fysiska och datalänksskikten. Pecket Loss och jitter-mätningar användes likaså. Vi antog att det fanns GSM tjänst på alla platser så att vi endast behövde göra uppg aende handover(t.ex. från WLAN till mobilt nätverk och inte tvärtom). Vi framkallade och testade en mekanism att starta handovern baserade på nätverksmätningarna.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Wireless Local Area Networks (WLAN), based on the IEEE 802.11 standard, have been widely deployed in recent years. Together with Voice over IP (VoIP), they enable a low-cost local area wireless telephony service. Users should be able to use this technology to reduce their costs by avoiding the use of cellular networks whenever they are within IEEE 802.11 coverage. Products that combine GSM and WLAN hardware already exist. Given suitable software to select which interface is to be used, users could roam between heterogeneous networks automatically. Therefore, the coverage of GSM networks with the cost savings of Voice over IP can be combined. Companies could set up WLAN networks to cover areas where users frequently are to reduce the cost of employee communications.

WLANs were initially developed for data traffic, and their ability to handle voice traffic was not of primary importance. Today the performance of IEEE 802.11 networks carrying voice traffic has been extensively studied [26, 29] with some researchers specifically examining handovers within an IEEE 802.11 network [33, 5]. The specific case of a handover between different networks, such as WLAN and GSM, is not specified in the IEEE 802.11 standard. The standard is only concerned with services that make use of the IEEE 802.2 Logical Link Layer protocol.

This thesis will focus on the challenge of implementing an effective handover between IEEE 802.11 and GSM networks in which the **perceived** voice quality is preserved. It should be done so as to complete the handover before the call quality deteriorates below a perceptible level, as the user moves out of WLAN coverage. The first section of this thesis introduces the problem and states overall goals of the project. Following this section, the background section briefly describes the relevant technical concepts involved in this study. The methodology section presents the tools and scenarios used for experimentation and the measurements section discusses the results. In the handover section we explain the design and tests of the

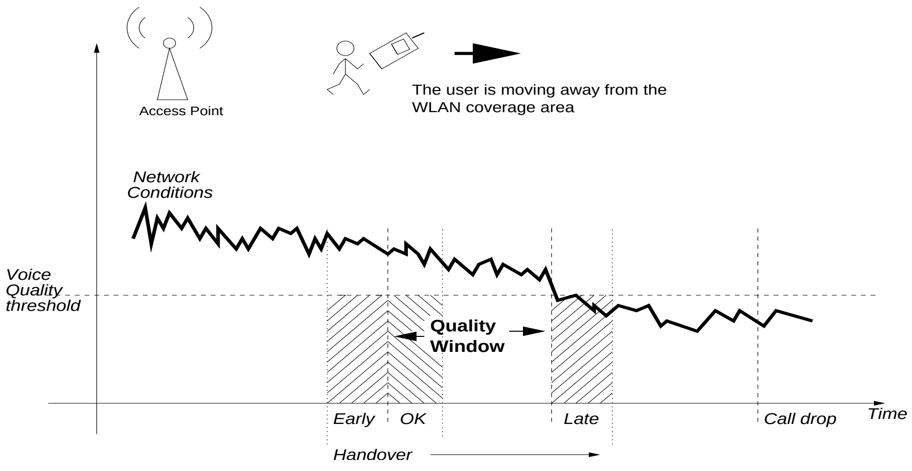

Figure 1.1: Time frame for a successful handover

handover trigger algorithm. This thesis concludes with some conclusions and future work.

1.2 Problem statement

Our main task is to design a handover trigger *based upon network conditions*. Five problem areas are listed below. The main issues are briefly explained for each one.

The first problem is related to prediction. The triggering function has to anticipate, based on current network conditions, when the voice quality is likely to deteriorate below some acceptable quality threshold. Additionally, we can describe a time window within which the handover procedure must be performed (see figure 1.1). As can be seen in the figure, a successful handover occurs when the algorithm initiates and completes a handover before the quality decreases below a threshold. If the handover is triggered early, unnecessary cellular usage costs may be incurred. As an example, consider a user walking around within WLAN coverage. Although he is moving, a handover to the cellular system would be categorized as early and unnecessary. When the handover is triggered too late, the user will perceive voice quality deterioration or, in the worst case, call disconnection. It is important to emphasize that disconnection is not an acceptable outcome for any of our proposed solutions.

The second problem is to identify and acquire the information needed to make the handover decision. A set of network parameters has to be measured in order to assess the current network conditions. Extracting these parameters from the incoming data stream should be independent of a particular hardware/software implementation.

A third issue is that the measurements should be performed with sufficient precision, granularity, and reliability to guarantee acceptable performance. At the same time it should remain feasible and economical. Therefore the problem addressed is to determine *which* parameters are useful in the prediction of network conditions. Other performance metrics could include cost savings for the end user.

The fourth problem is to develop a function that combines the results of these parameters according to their influence in the prediction process, i.e., the parameters are weighted to minimize the probabilities of incorrect handover decisions.

Finally, an effective handover algorithm should be practically evaluated with regard to preserving call quality. The fifth problem is to determine a methodology for experimentation and testing under different environmental and network conditions.

1.3 Solution scope

In current wireless systems, most handover algorithms make use of signal strength measurements for handover decisions. In general, as the person walks away from a base station or Access Point, the signal strength of the current connection decreases until it falls below a threshold or until its value is lower than that of neighboring base stations. At that moment the handover should occur. In GSM systems[18] when a call is active the mobile terminal measures the signal strength from all the current operator's base stations detected in the surrounding area. An estimation of the bit error rate is computed and forwarded to the current base station controller. This estimation is done by the GSM terminal every 480 ms. Based on the received data and other measurements, the Base Station Controller (BSC) determines when a handover is scheduled. This is often referred to as Mobile Assisted Handover MAHO.

In the case of IEEE 802.11 networks, the responsibility for making the handover decision lies within the mobile terminal. Within this standard handovers are defined only at the link layer, for handovers between access points in the *same subnet*. The process is based on the received signal strength that mobile terminals measure either actively using probes, or passively using received traffic or control frames. Further details of the process are not standardized and are implementation dependent [33].

Our hypothesis is that additional parameters may improve the accuracy and reliability of the handover decision. Examples of parameters are location information, noise or interference, packet loss, or number and pattern of link layer retransmissions. Higher-layer factors such as cost, terminal speed, or even user preferences could also be taken into consideration. It might be possible to deploy an external server that monitors the terminal's communications and makes a handover decision based on quality reports sent periodically by the terminal.

1.3.1 Goal of the project

The goal is to predict sufficiently in advance **when** to make a handover based upon the conditions of the network. This will be done using a set of parameters that accurately predicts the moment of an excessive

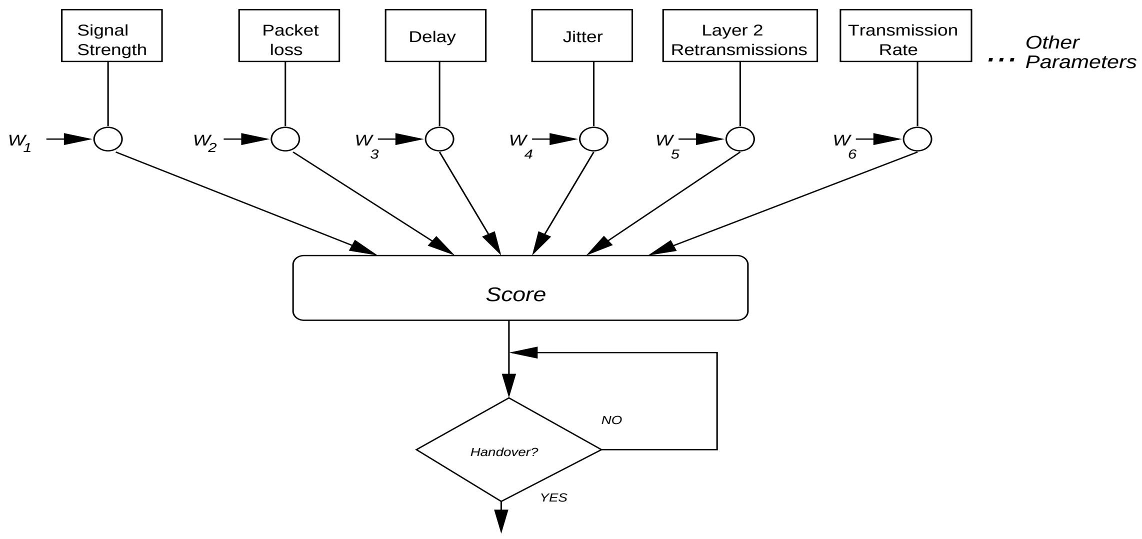

Figure 1.2: Overview of a function that predicts voice quality

call degradation. Our proposed function will combine the parameters into a "handover score" that indicates when a suitable handover should be triggered. The function should predict when the network conditions will fall below a predefined level, and trigger a handover to the cellular network for the current call. In figure 1.2 we show a generic flow chart of an algorithm that generates a handover score.

1.3.2 Limitations and assumptions

In the following two sections we list some of the limitations and the assumptions made during the course of this project.

Limitations

The first limitation is our selection of the parameters for the handover trigger, we have selected network conditions that can be determined from the voice data stream. The use of external agents in the network is avoided so the implementation can operate entirely within the mobile terminal. Secondly the mobility of the user will be limited to a walking speed (i.e., less than 2m/s). Finally the design should be as independent as possible of coding schemes, hardware platforms, and software implementations.

Assumptions

It is expected that a mobile terminal with both IEEE 802.11 and cellular interfaces is available. A telephony application is assumed to reside within the handset. We assume that this handset is programmable. Since our goal is to save costs, the cellular interface is assumed to be either disabled, or in low power mode *unless a handover is needed*. We also assume that the call is continued when leaving WLAN coverage or when poor WLAN conditions arise.

It is assumed that the cost of using WLAN infrastructure to make voice calls is zero or very low. Furthermore the cost of GSM service is higher and charged per minute of connectivity. We assume that most of the calls will be terminated within the same LAN, i. e., between employees of the same company. Calls using the public PSTN (Public Switched Telephone Network) or cellular networks, outside the LAN, is an additional service and should be used as little as possible [10].

Chapter 2

Background

The technical part of this project involves concepts related to the IEEE 802.11 standard, Voice over IP (VoIP), and quality assessment. In this section we provide a brief description of these concepts.

2.1 IEEE 802.11 Wireless Local Area Networks

The physical as well as media access and control (MAC) mechanisms were defined by the IEEE body within the 802.11 working group. The current versions of the standard are 801.11b, with data rates up to 11 Mbit/s, and 802.11a and 802.11g that can reach up to 54 Mbit/s. According to the base 802.11 specification, the network elements are the Wireless Stations (STAs), the Distribution System (DSS) that interconnects the radio cells, and the Access Point (AP), which connects wireless stations via the radio interface to the distribution system.

At the physical layer the standard defines the frequencies, modulation, data rates, and coding schemes. The MAC layer defines the frame structure and contention method. All 802.11 networks use a collision avoidance method, in which stations have to sense and optionally reserve the media whilst transmitting. Additionally, for large frames a Request To Send / Clear To Send process is defined (RTS/CTS). The frame size threshold for RTS/CTS can be set by the user. The frame size for voice traffic is too small for the RTS/CTS mechanism to be used.

The 802.11 architecture builds upon a Basic Service Set (BSS), which consists of interconnected wireless stations, either independently in an *ad hoc* fashion (called an Independent BSS) or through the AP (called an Infrastructure BSS). Further details on 802.11 networks can be found in [11].

2.2 Factors that affect voice quality

In the following section a short background of the parameters that influence voice quality is presented. In addition basic VoIP concepts that are of relevance to the project are included.

2.2.1 Effects of data transmission

In modern fixed line telecommunications systems, the analog voice signal is sampled and coded producing a fixed-rate bit stream. As with most communications systems, this data can be corrupted, lost, or delayed. These impairments have an effect on the final decoded signal and consequently on the quality of the perceived sound. Evaluating these degradations and their effects on the human listener is difficult. The best way to determine the effects for voice transmission is by a panel of people listening and assigning a subjective evaluation to the received speech.

However, using panels of professional listeners is time consuming and expensive. Hence, the usual method is to automatically measure network conditions based on metrics that influence the perceived voice quality: losses, delay, and jitter. Higher layer effects such as echo, background noise, and the selected voice coding can also affect the perceived quality.

Losses in packet networks occur when a frame or packet does not reach its destination. In real-time multimedia applications packets that arrive too late are treated as if they were lost. Delayed packets cannot be assembled on time to the user interface. Packets are also discarded when some part of their payload has been corrupted as determined by an error detection mechanism. In voice transmission a concealment technique can be used to reduce the perception of losses. One technique is based on the repetition of a previously received sequence during the gap left by a lost packet.

Delay is the time needed for data transmission and reception. For a high quality voice communication the overall delay should be as low as possible. The International Telecommunications Union ITU defines a threshold of 150 ms of end-to-end delay[16]. However, in real life the delay is based upon the type of voice communication - for example interactive versus simplex. Time spent processing, buffering, transmitting, propagating, receiving, transmission, and coding/decoding all contribute to the total delay.

Jitter refers to the variance in the arrival delay of the individual packets. Normally this effect is mitigated by the use of playout buffers that temporarily store the frames and absorb the time differences. If the amount of jitter is greater than the capacity of this buffer, the packet will be considered lost - even if it does eventually arrive, it is too late to play.

2.2.2 Degradation due to coding

Prior to a digital transmission an analog voice signal is sampled, quantized, and converted into a digital stream. An example is the 64 kbit/s data stream used in regular fixed line telephony. This data can be further compressed using techniques known as perceptual voice coding. This technique uses properties of the human voice and ear to reduce the amount of information, thus reducing the bandwidth needed for the transmission. The use of a voice coding technique degrades the voice quality, and its effects are different for each algorithm. Usually a tradeoff is made between the available bandwidth and the desired quality in order to select the algorithm used for the coding/decoding process.

For this project we will consider G.711 coding with μ-law, which maps a 16 bit sample from the input audio device to an 8 bit code with no further processing. The packetization interval we will use is 20 ms. This type of coding is used in the public switched telephony network. It is also relatively insensitive to packet losses since each packet is independent of those preceding and following it.

2.3 Voice over IP

Voice over IP (VoIP) is a technology that emulates a circuit switched voice channel using the Internet Protocol (IP), a packet switched data technology.

VoIP makes use of a set of protocols to handle voice data and signaling information. Most applications consist of a coder/decoder (CODEC) for the initial processing. Then they transmit the encoded samples using the Real Time Protocol (RTP) on top of an IP connection. RTP is defined in RFC 3550 [28], and provides for correct playback of multimedia information. It uses sequence numbers and timestamps to determine if the packets arrive on time and to play them in the correct sequence. For signaling, different protocols are defined, for example ITU H.323 or the Session Initiation Protocol SIP [27].

A control protocol is defined for the exchange of statistical information concerning the quality of the RTP flow. The Real Time Control Protocol (RTCP) describes the report formats for the sender and receiver, within the RTP packet. The information in RTCP was proposed as a payload extension in RFC 3611 [9]. It provides detailed measurement parameters such as signal and noise levels, loss and discard rates, delay information, and quality estimations. Further information about VoIP is found in [22].

2.4 Voice Quality Assessment

The International Telecommunications Union (ITU) Recommendation P.800 [14] defines the procedure for *subjective* voice quality assessment. Factors such as room dimensions, noise conditions, the set of voice samples, and the characteristics of subjects are defined. The recommendation specifies a five-point scale that all listeners should use to evaluate their perceived quality of the speech. The mean of these five-point values are referred to a Mean Opinion Score (MOS).

A popular algorithm for objective voice quality assessment is PESQ [17]. PESQ's implementation is based on a psycho-acoustic model of the human auditory system. A sample of both original and degraded signals are fed to the algorithm to compute a final ”quality score”. The need for a sample of the original signal does not allow its use in real time systems.

2.5 Related work

We present in this section the state of the art in handovers from IEEE 802.11 WLANs to cellular networks. We indicate the relevance of the chosen work to our own.

2.5.1 Handover between WLAN and cellular

At the Swedish Institute of Computer Science (SICS) a series of studies were carried out by Marsh et al. [21]. A trigger for WLAN to GSM handover was proposed based on quality estimation. A number of network parameters were identified and evaluated for triggering a handover: signal strength, losses, jitter, and transmission rates. The suitability of each parameter was determined. Based on individual tests for each parameter they assessed the accuracy, granularity and influence in perceived quality to create a “voice quality score”. A handover prototype was implemented and evaluated in a simple experiment. It resulted on a correct decision rate of 83%. Out of 100 experiments, 68 resulted in true positives and 15 in true negatives. Of the remaining 17 experiments, 7 were false positives and 10 resulted in false negatives. These studies identified useful parameters for handover, and presented a method for achieving seamless vertical handovers between such networks. The overall goal of this work was *triggering* the handover and not prediction, hence information from upper layers and other factors that could influence the decision (location information, cost, or economic issues) were not considered.

Hasswa et al. [13] proposed a handover decision function for roaming between heterogeneous networks such as WLAN and GSM; they identified as the factors that influence the handover decision: cost, security, power consumption, user preference, and speed of the user. A function that estimates the network quality based on these parameters was described. In this function a set of weights were assigned to each parameter, according to the kind of network to which the user is roaming. Tradeoffs that influence the relations between parameters were also explained. They conclude that such a model was feasible and can be applied, but it was largely a theoretical study and did not define how to accurately obtain the parameters described. Furthermore, the problem of predicting network conditions in advance was not addressed. However, this model could be applied providing a way to overcome these issues based upon further experimentation.

Hung Ju Tze [31] in his thesis addressed the same goal (i.e., of seamless handovers for voice calls between IEEE 802.11 WLAN and GSM). After the analysis of a business model, he experimented with network parameters such as capacity and interference. He implemented a solution for handover between IEEE 802.11 WLAN and GSM. He used the call forwarding function already implemented for a handheld computer. He concluded that this implementation was technically feasible and economically viable mainly depending on the cost of the user terminal. This project aimed to develop a working prototype for the handover. Although he determined a threshold for a handover decision based on signal strength and packet loss, defining a handover trigger was out of scope of his experiments. The handover decision was performed manually. The perceived voice quality was not measured.

2.5.2 Commercial products

As dual mode handsets have become available, recent developments address the problem of voice calls using 802.11 networks based upon commercial products. Some of them are briefly described in this section.

A group of companies have together developed a technology called Unlicensed Mobile Access (UMA) [32]. In UMA a user with a dual mode Wi-Fi[1] - GSM phone can make calls with seamless handovers between the two technologies. In the UMA architecture a new network element is placed between the GSM network and the Internet. This node provides a way to transport GSM signaling and calls over IP connections. The approach used in UMA is different than the situation we consider since the cellular operator is always in control of the call and the user needs to order the service from this operator. New protocols and elements are introduced in the network as well. UMA technology is under development and supporting vendors such as Nokia, Alcatel, and Motorola, are currently performing trials with operators in the United Kingdom and Denmark. The development of UMA is a responsibility of the 3G Partnership Project 3GPP since 2005.

In Norway a service similar to UMA that offers seamless handover between Wi-Fi hot spots and GSM is now being commercially offered by the operator HELLO [6]. The system is based on special software installed on a PDA or smart phone that logs into a central server that keeps track of the locally available links and triggers the handover. The software was developed by the Irish company Cicero networks [24].

Finally, Optimobile AB [4] offers a corporate telephony solution that integrates a VoIP server and a software client installed in a dual mode IEEE 802.11 - GSM terminals. This technology enables calling at low cost whenever IEEE 802.11 coverage is present. It includes an algorithm for handover of calls between a WLAN and cellular, whenever the person leaves the WLAN coverage. The server also integrates data services to be used with the platform. Although very relevant to the topic of this project, the handover algorithm has not been evaluated, and further technical information is not available.

[1] Wi-Fi is a branding and interoperability initiative for IEEE 802.11b.

Chapter 3

Methodology

This chapter discusses the methodology used to collect and analyze the experimental data. First of all, we describe the approach that was used. Second we present the selected hardware and software tools. Third we define the measurement scenarios, and finally we outline the structure of the software implementation used for all experiments.

3.1 Approach

We chose an experimental approach aimed to obtain accurate information that could be used to model a common handover scenario (IEEE 802.11 to cellular). All experiments were based on measurements that extract parameters of the IEEE 802.11 data stream and the RTP session of a VoIP call. We first observed the information provided for each parameter. Then we aimed to determine the significance of each parameter with regard to a handover decision. Finally, our purpose was to integrate this information in order to provide a detailed representation of network conditions.

We used software tools that allow extraction of information in real-time. These tools are described in section 3.3. Based on preliminary experiments we selected a number of physical and link layer parameters that provide information on network conditions. For each selected parameter we used statistical tools to process each result. The results obtained were used to design a "handover score" algorithm. This algorithm shows the decrease in perceived voice quality and thus the moment to trigger the handover.

3.2 Hardware used for the experiments

The WLAN hardware interface used was the Netgear W511T Cardbus Wireless Adapter. This card is based on the Atheros AR5005GS series chipset with firmware version 4.3 [8]. It is compliant with IEEE 802.11 b/g standards and incorporates a proprietary technology that increases the transmission rate up to 108

Mbit/s. These proprietary features were deactivated during our experiments, and the card was configured to work in IEEE 802.11b mode only. This standard was chosen in order to compare our results with previous studies [21] [31].

The card has a built-in omnidirectional antenna with 6dBi gain and a maximum transmission power of 18 dBm. The wireless card was controlled by an open source experimental driver with advanced administrative tools as described in section 3.3.2.

All measurements were performed using an HP ze2000 laptop computer with an AMD sempron 1800+ processor. The other end of the wireless link was a Linksys WAP54G Access Point (AP). For this project only the basic functionality of the AP was used.

3.3 Software used for the experiments

3.3.1 Operating system

The laptop computer was running Fedora Core 4, a Linux distribution with kernel version 2.6.17. However, all measurement software should work with small changes in other Unix-like environments.

3.3.2 Madwifi driver

Madwifi is an open source driver written for Atheros chipsets in Linux environments [2]. For our experiments we used version 0.9.2 of the Madwifi driver. Madwifi includes wireless troubleshooting features such as the support for monitor and AP modes, allowance for customization of the data rates, transmission power, power save modes, beacon intervals, and encryption. We were primarily interested in a tool included with Madwifi. It is called *athstats* and presents a list of link layer information that is read directly from the Atheros hardware.

For this project the code of the *athstats* program was modified and recompiled to adjust to new measurement conditions and to allow the integration with the VoIP tools we used.

Details of the parameters measured with Madwifi

After a number of preliminary tests, only those parameters that were related to the WLAN traffic were selected. In total, based on our experiments we selected 4 out of 35 possible parameters. A brief explanation of their meaning according to the IEEE standard and frame format is presented below.

1. *Transmission on-chip retries*: The error detection mechanism used in the wireless hardware implements two internal counters labeled *long* and *short retry count*. These counters are incremented whenever a frame fails to be transmitted due to a busy media, or at every retransmission until a

successful ACK message is received. The difference between long and short depends upon whether the size of the frame is below the threshold for an RTS/CTS exchange. This is customizable by the user, in the Atheros hardware the RTS/CTS feature is turned off by default.

2. *Reception failed because of bad CRC*: The generic 802.11 MAC frame format used for data and control messages includes a trailer field of 4 bytes labeled "FCS". It is a CRC algorithm for error detection. This flag indicates a frame reception that did not pass the CRC check.

3. *Transmission RSSI of last ACK*: The Received Signal Strength Indicator (RSSI) is defined in the IEEE 802.11 standard as a value between 0 an 255 that indicates the signal strength of the received packets. Positive values indicate better signals. It is generated and used internally by the hardware and is not included in the transmitted frames. The range is vendor dependent: A manufacturer defines an *RSSI-max* value which according to measurements in [19] can be mapped to -10 dBm, an average reception level for a short distance. In the Atheros chipsets the *RSSI-max* value is 60, and is converted to dBm by subtracting 95 from the RSSI[19].

4. *Reception PHY error summary count*: The physical layer of IEEE 802.11 includes a bit sequence known as *preamble* used for synchronization purposes. It carries information on the modulation scheme, frame sizes, and data rates in order to properly read the frame. Any failure in decoding this information is reported as a *PHY error*, and the frame is not passed to the MAC layer. If the CRC is verified, the frame has already passed the PHY verification. Therefore CRC and PHY error counters indicate errors at separate layers [7].

3.3.3 Sphone

Sphone [12] is an application developed by Olof Hagsand that implements a VoIP sender and receiver. It is an implementation of the IETF protocols RTP and RTCP. Sphone was used to send voice traffic for our experiments. The software stores sequence numbers and timestamps of the RTP header allowing the calculation of packet loss and jitter.

We selected Sphone due to the availability of the source code. It allowed us to gather statistics and implement a calculation in real time. The RTP sequence number can be used to determine the packet losses by detecting a missing number in the list of received packets. We used timestamps from the sender stored in the RTP header and the local arrival time of the packet to calculate jitter.

Sphone's receiver application was modified to allow integration with *athstats*. When a packet arrives, the Sphone application calls the *athstats* function and the RTP and IEEE 802.11 parameters are saved in a log file.

3.3.4 Wireless Tools

IEEE 802.11 support for Linux is built upon the software called *Wireless Extensions* [30], a open source project that includes the API and libraries for networking support. A standard set of applications based on these Wireless Extensions are called *Wireless Tools*, they allow the user to set parameters and gather information from the wireless interface.

The application *iwconfig* is part of these tools. It outputs interface statistics such as current network identifier (BSSID), IEEE 802.11 channel, transmission rate, transmission power, and signal-to-noise levels reported by the card.

Details of the parameters measured with Wireless Tools

The code in *iwconfig* was also modified to extract the following parameters:

1. *Transmission Rate*: In 802.11b, the data rate can be dynamically adapted according to the network conditions. The supported rates are 1, 2, 5.5, and 11 Mbit/s. They are advertised by stations and APs using probes. This value is included a management frame field.

2. *Transmission Power*: Transmission levels are shown in dBm. This value can be set by the user in some APs or wireless cards.

3. *Signal and Noise Levels*: For the Atheros hardware, this indicates the value in dBm for the signal strength and noise levels. The signal level can be obtained from the RSSI.

3.3.5 Software modifications

We modified the sphone receiver source code *rcv.c*. If the logging option of *rcv* is activated, information on every received packet will be read and logged to a text file. Within this routine we included a call to *athstats* and *iwconfig* to capture the relevant IEEE 802.11 parameters. File interaction and function names are shown in figure 3.1. We logged the information on a packet-by-packet basis according to the log function implemented in Sphone. However it is foreseen that the IEEE 802.11 parameters can be sampled more frequently if necessary.

3.4 Experimental description and network configuration

In this section we describe the technical details of the experimental setup. We sent RTP traffic between one computer on the wired network and a mobile wireless host. We configured the wireless card to work in 802.11b mode and the RTS/CTS mechanism was disabled.

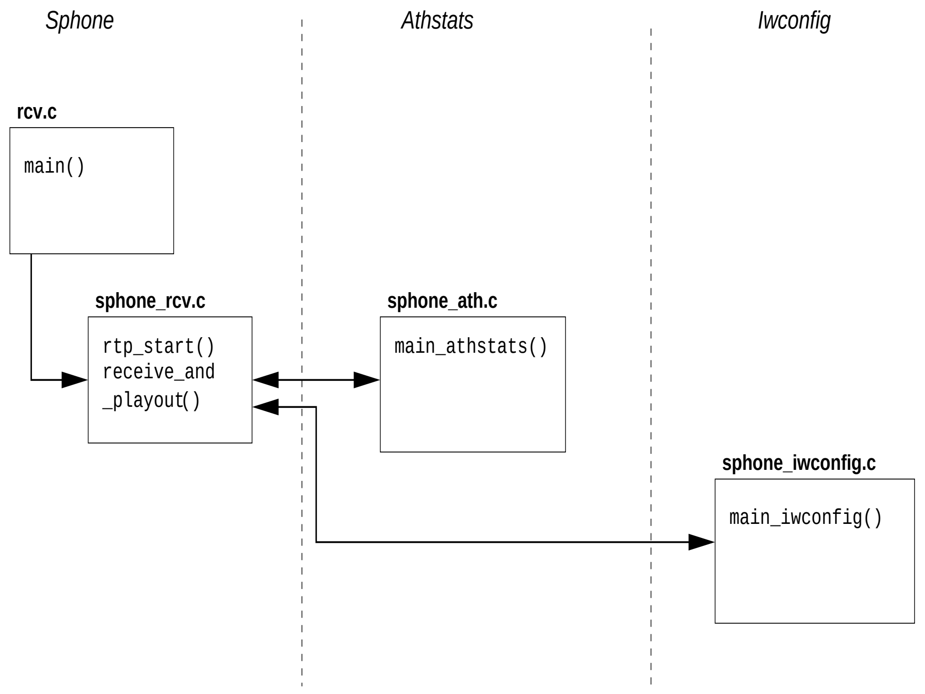

Figure 3.1: Block diagram of the software implementation

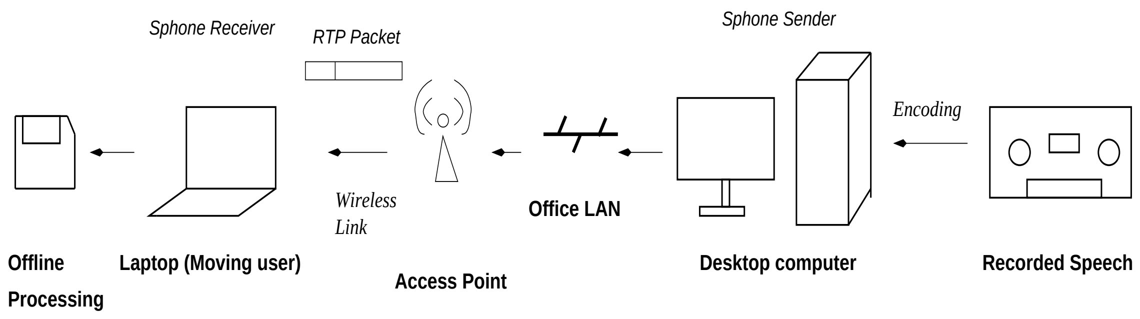

Figure 3.2: Diagram of the experiment

In Figure 3.2, we show a diagram of the experiment. *Sphone* uses a prerecorded speech sample extracted from a file and encodes the samples as RTP packets. The encoding method is μ-law, 8000 samples per second and 8 bits. Sphone encodes 20 ms of speech at a time. The μ-law coding scheme is defined by the International Telecommunications Union (ITU) for voice calls in the public network [15]. Packets can be sent in two modes: simplex (one-way) or duplex. However, the logging mode that we used was available on Sphone only for one-way transmissions.

We emulated a one way voice session between the two computers. *Sphone* was used to send traffic as described above and *athstats* measured the IEEE 802.11 and VoIP metrics. The Sphone sender was running on the static, wired host. This host was a Linux desktop computer connected to the office LAN. The Access Point was connected to the LAN as well. Sphone was configured to send RTP packets to the IP address of the wireless host. Upon arrival at the wireless station the modified sphone receiver logs to a text file the

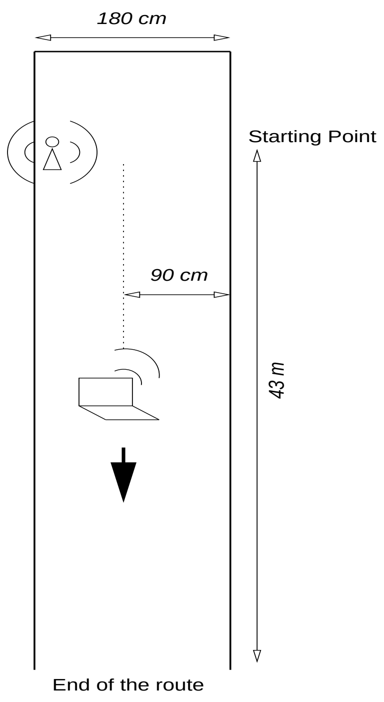

Figure 3.3: Indoor test site with no obstacles

contents of the RTP header. We used the sequence number and time stamp for our calculations. The IEEE 802.11 parameters from *athstats* were also recorded. An example of the log file format is given below:

```
[root@dhcp01 ~]# tail athdata.txt
lretry crc  phy    rssi Txrate Loss   RTPtime  Localtime
0      0    0      35   11     0      6.137481 6.147610
0      0    5      36   11     0      6.157486 6.163613
0      0    0      36   11     0      6.177490 6.191612
0      0    0      36   11     0      6.197494 6.207625
0      0    1      35   11     0      6.217483 6.224028
0      0    0      35   11     0      6.237518 6.241205
0      0    2      35   11     0      6.257492 6.263688
0      0    0      35   11     0      6.277496 6.281172
0      0    2      35   11     0      6.297485 6.303742
[root@dhcp01 ~]#
```

Example of text file with recorded data

The first three columns show the measured values from the lower layers: long retry counter, CRC error and physical error counters. The following two columns show the RSSI (a value from 0 to 60) and the transmission rate in Mbit/s. The sixth column shows the lost packets based on the RTP sequence number. In the last two columns the time stamp of the RTP packet and local time are recorded.

3.4.1 Preparations and calibration

We conducted a series of measurements in the office corridor shown in figure 3.3. The goal was to emulate real conditions of a voice user. Therefore, we wanted to assess the radio conditions including any interference. The first measurement consisted of scanning for neighboring wireless networks. Note that our environment is a modern high-tech office on the 6th floor and many of the surrounding companies use IEEE 802.11 networks. We used Kismet [1], an open-source packet capture program, to detect these networks. We detected that channel 1 was the busiest while channels 2 and 12 were less used.

Second, we tried to detect the presence of sources of interference. The laptop was placed at four different stationary locations. We recorded the number of CRC errors for a 20-second Sphone transmission at each location. This chosen period covers four handover execution intervals. We considered a handover execution time of 5 seconds. Initially we selected for the testing channels 1 and 6 of the IEEE 802.11 standard. The results and final selection of the channel for the measurements are given in section 4.

3.4.2 Description of the selected location

The measurement locations are intended to emulate the environment of an average user. We selected a typical office corridor for the experiments.

The measurement scenario involves a user moving away from the access point. As can be seen from figure 3.3, the AP is located close to the end of the corridor. It was placed above an office door at an open space by the end of the ceiling. We walked down the corridor from the AP's position towards the main entrance, covering a distance of 43 meters. The laptop was placed on a wooden trolley (with metal frame) 78cm high. We moved the trolley at a constant walking speed. Both the card and AP antennas were horizontally oriented. The antennas were pointing upwards as in a real case.

Chapter 4

Measurements of IEEE 802.11 parameters in a handover experiment

The purpose of this chapter is to show the results of the measurements of IEEE 802.11 network parameters. We have individually studied each result and selected the most representative results for each parameter. A more detailed analysis is included in the section 5.3.2.

4.1 Preparatory experiments

In the preparatory experiments, we monitored the number of CRC errors on three different IEEE 802.11 channels in order to detect interference. This parameter was chosen because the use of signal-to-noise ratio was not possible. The WLAN card did not provide an accurate value of noise. The first location was at four meters from the access point and the results are shown in figure 4.1. The laptop was very close to the AP and the error counter recorded up to 16 errors. This is the normal behavior and it should not affect the overall performance. This is shown in figure 4.1. The results were similar on the two channels measured.

A second location was chosen in the same corridor 16 meters away from the Access Point. In figure 4.2 we show the CRC errors registered at this location. In general we did not observe a significant change from the previous location. Only a small increase of errors in channel 6 was measured. However we did not consider this significant.

In a third location at 25 meters the CRC errors increased. 30 to 40 errors were recorded in some of the 20 ms intervals as shown in figure 4.3. The last location at 32 meters showed a similar behavior In figure 4.4 a higher number of errors were registered for channel 6.

The higher interference found at locations 3 and 4 was probably caused by a computer room close to the antenna. However, as it was not possible to measure this interference, we decided to conduct the

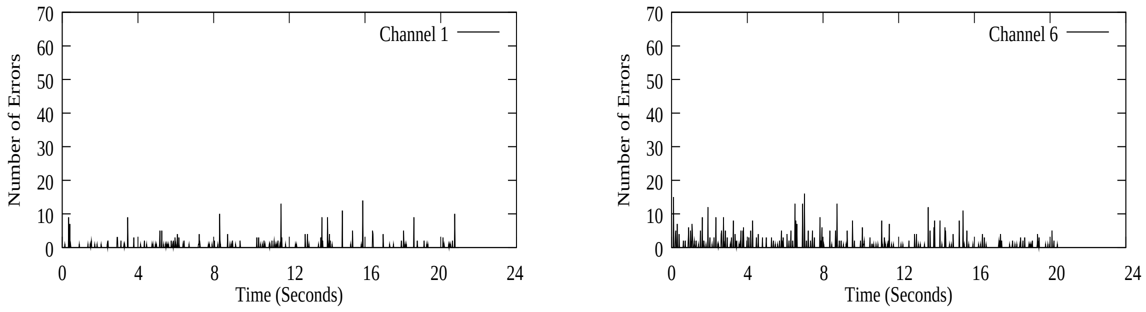

Figure 4.1: Number of CRC errors value per VoIP frame as measured at a location 4 meters away from the AP

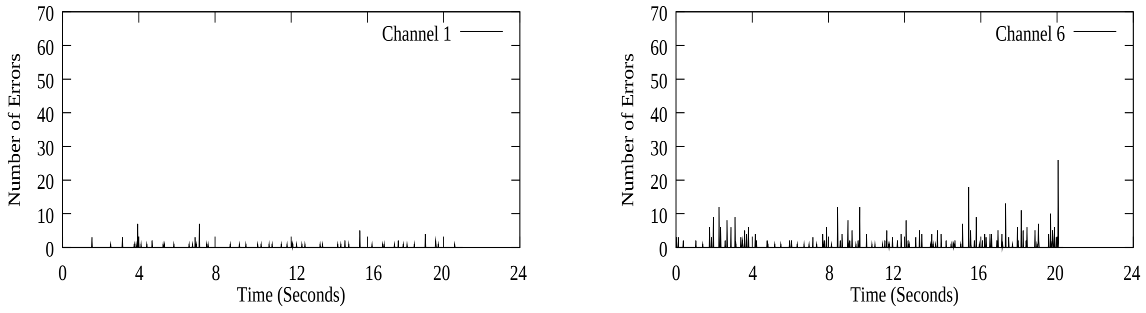

Figure 4.2: Number of CRC errors per interval of 20 ms, 16 meters from the AP

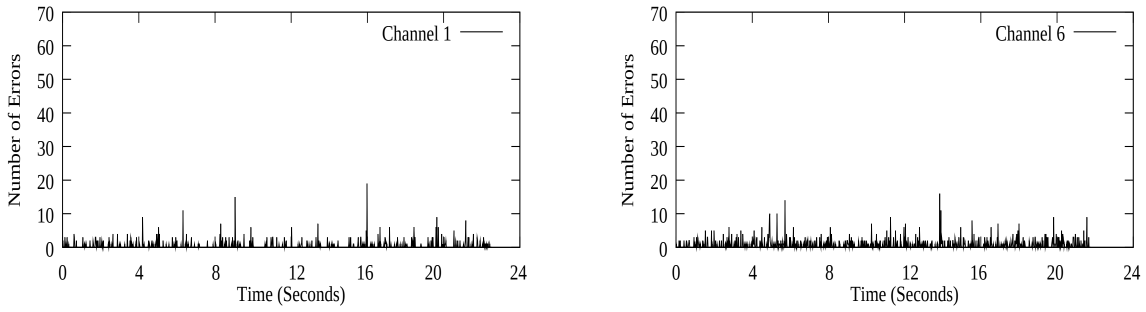

Figure 4.3: Number of CRC errors per interval of 20 ms, 25 meters from the AP

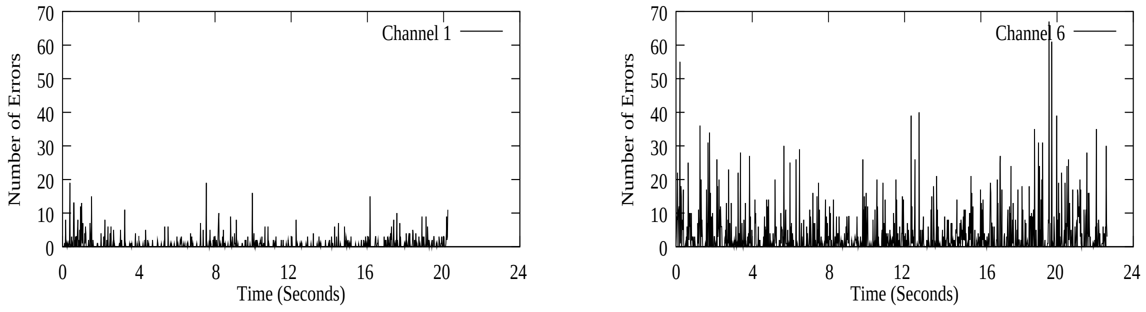

Figure 4.4: Number of CRC errors per interval of 20 ms, 32 meters from the AP

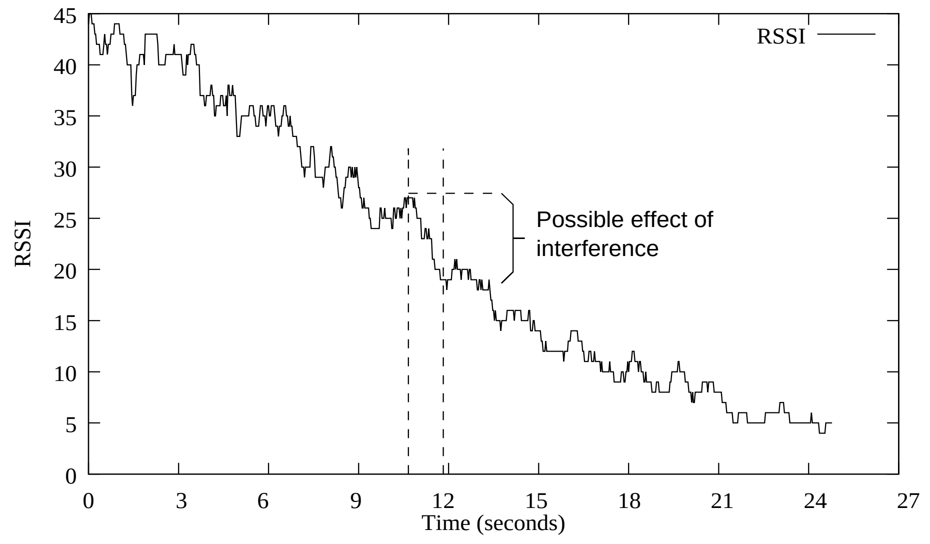

Figure 4.5: Measured RSSI for a moving host in the office corridor

measurements here. This situation could also occur at any office environment. We concluded that channel 1 was the most suitable for the measurements. The results indicated less interference than channel 6, and it is more used in real networks.

4.2 Individual results for each parameter

In this section we describe the measurements made for the IEEE 802.11 and RTP parameters. First we discuss the measurements at the physical and link layers, then the results at the IP and RTP layers.

4.2.1 Link layer parameters

In figure 4.5 the RSSI value shows the value of the signal strength. The laptop was moving away from the AP. At the beginning the value is maximum and it starts to decrease. The minimum value of RSSI at the the furthest point is 5. The graph shows a decreasing trend with small random fluctuations.

The area within the dashed lines in the figure shows a drop of approximately 5 dB that was measured while passing in front of the computer room. We did not consider that this loss was significant for the received signal. This idea was confirmed when we looked at the recorded sound file. The perceived quality of the voice was not affected.

In figures 4.6, 4.7, and 4.8 we show the results for the number of physical errors, CRC errors, and long retry counters. These values are inversely proportional to the RSSI. They increase more rapidly when the RSSI is at its lowest value. As expected, the rate of physical errors was higher than the CRC since they are the first filter for corrupted frames. The retry counters only increased when the RSSI level was below the value of 10.

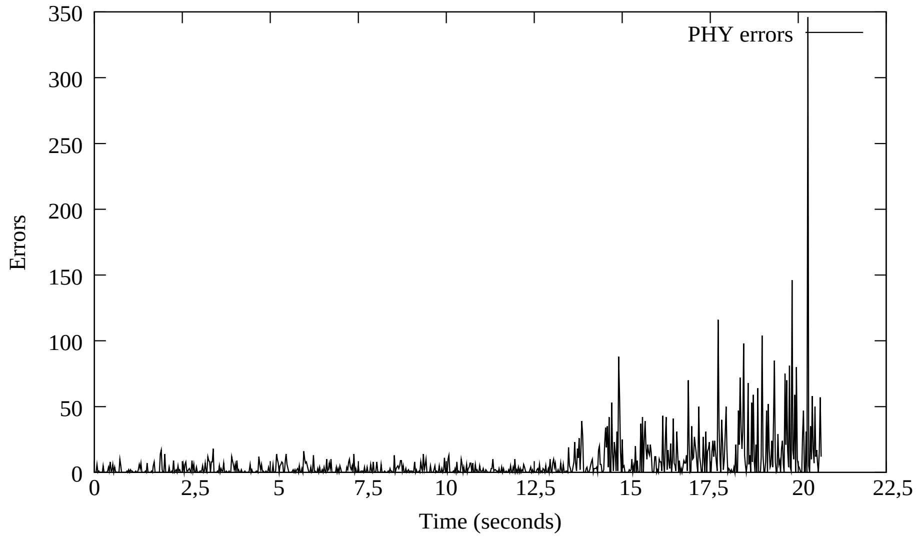

Figure 4.6: Physical error counter value per VoIP frame as measured for office corridor

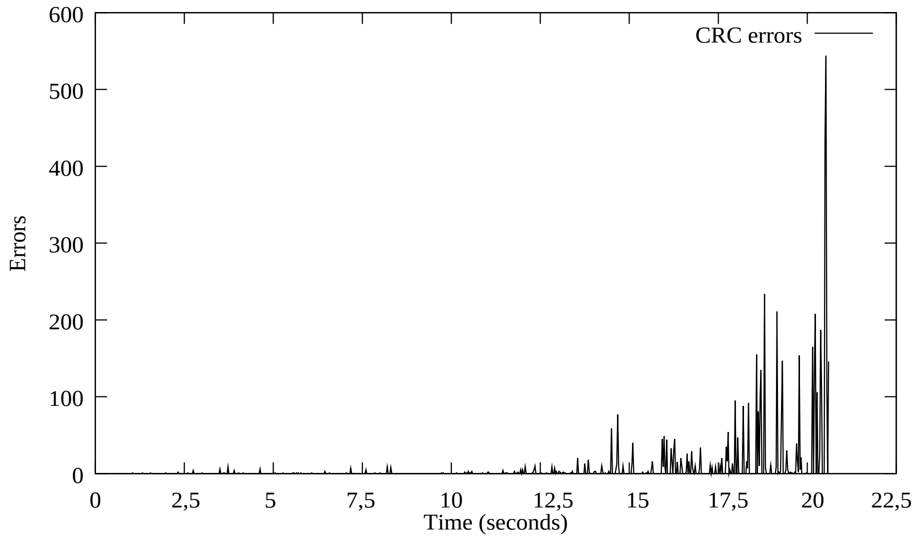

Figure 4.7: CRC error counters value per VoIP frame as measured for the office corridor

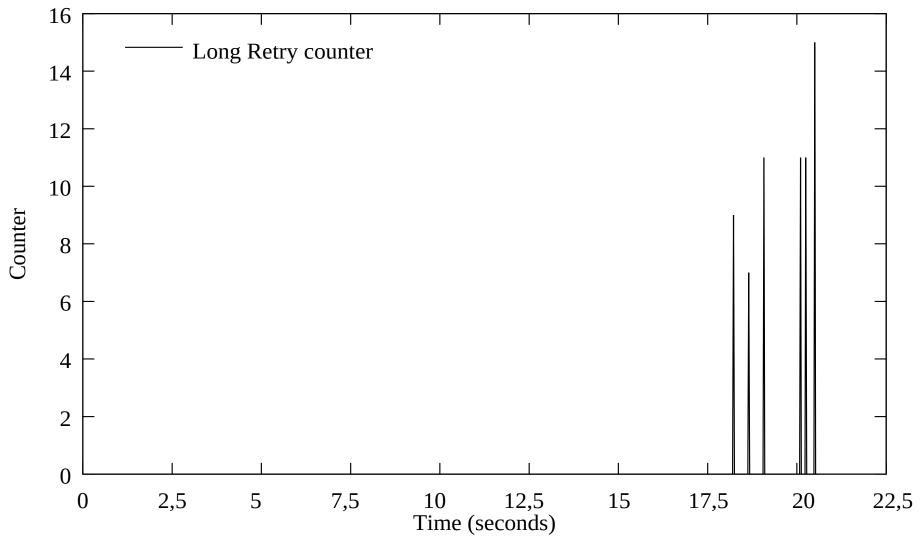

Figure 4.8: Long retry counter value per VoIP frame as measured for the office corridor

The physical error counters start to increase their level above negligible noise values earlier than CRC or retry counters. Their values have also more regular shapes. Peak values of physical errors are measured at the same points at every measurement. CRC errors increase later and more rapidly with higher variance between measurements. At the end of the corridor both parameters show similar values. In conclusion, physical errors show a more predictable trend that can be used to predict network conditions. However we consider that a deterioration on the quality of the connection can only be shown by a combination of these two values.

Retry counters show little activity compared to the previous two measurements. One possible reason was that the application was only receiving voice packets. Retry counters are an indicator of the transmitted frames. Our host only transmitted acknowledgments and management frames at the MAC level. The lack of results on retry counters can be caused by the reduced number of transmitted frames.

4.2.2 Changes in transmission rate

The transmission rate adaptation algorithm in the Madwifi implementation is described in [20]. The transmission rate does not depend on the received signal strength, it is rather adapted based on the frame loss results. It is increased one step if the number of successful transmissions rises above a threshold. It decreases after a number of missed ACK packets and the thresholds can change dynamically. It was not possible to corroborate this algorithm. The hardware used did not present accurate results for frame losses.

We monitored changes in the transmission rate for 802.11b mode. We observed that the value does not change if the rate at the beginning of the measurement was negotiated at 11 Mbit/s. As it can be seen in the figure 4.9 if the initial data rate was lower than 11 Mbit/s (1, 2, or 5 Mbit/s), it returned to 11 Mbit/s as soon

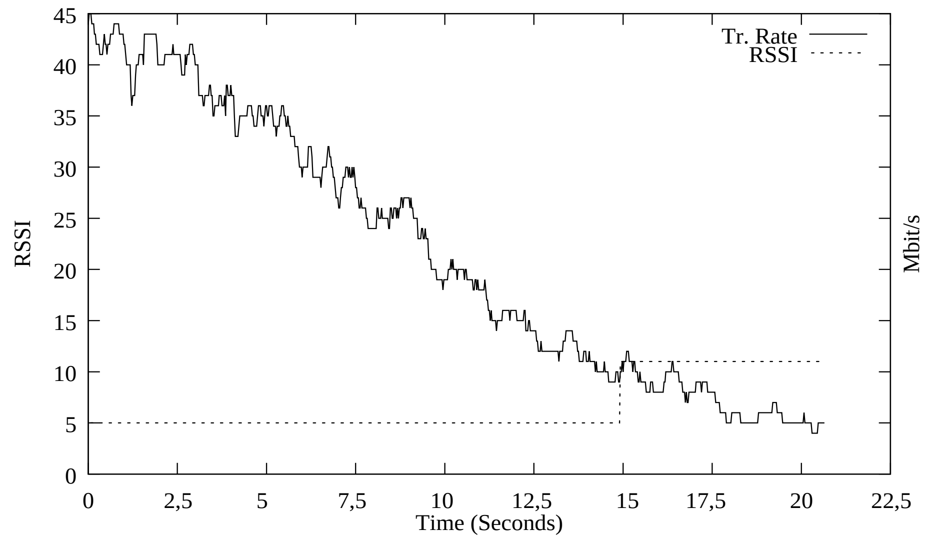

Figure 4.9: Transmission rate and RSSI measured for for the office corridor

as the RSSI fell below a threshold. The figure shows an example of one measurement in which the data rate increases from 5 to 11 Mbit/s when the RSSI number was below 10. However, we expected the rate to decrease because the link must become more robust. This can be caused by a bug in the algorithm that adapts the transmission rate in Madwifi. This behavior has been reported to the driver's designer group.

4.2.3 RTP layer measurements

We measured the packet losses based on the sequence numbers and timestamps received in the RTP header. In figure 4.10 we show the measurements of packet losses compared with the RSSI. As can be seen, in the first 12 seconds no losses were detected. The short distance and the absence of obstacles between the host and the AP ensured a good reception. When the distance increased, intermittent losses of 1-2 consecutive packets were recorded. In the last three seconds the rate of loss rises considerably. Up to 15 or 20 consecutive packets were lost.

The impact of the packet losses has to be determined by using a method for quality estimation. We compared our results with similar measurements by Marsh et al. [21], which indicated that bursts of 10 consecutive losses or less have no effect in the listener, we exceeded this threshold only during the last two seconds. The quality of the received voice should have been affected.

The comparison of packet losses with the CRC or physical errors is shown in figure 4.11. Physical errors show deterioration after 12 seconds while packet losses do not increase until second 17. Therefore the physical and CRC counters can be useful as an early warning of the deterioration of network conditions.

A plot of the jitter is shown in figure 4.12. During the first part of the measurement its value did not exceed 1 millisecond. At the end of the measurement it only increases until 3 milliseconds at the highest

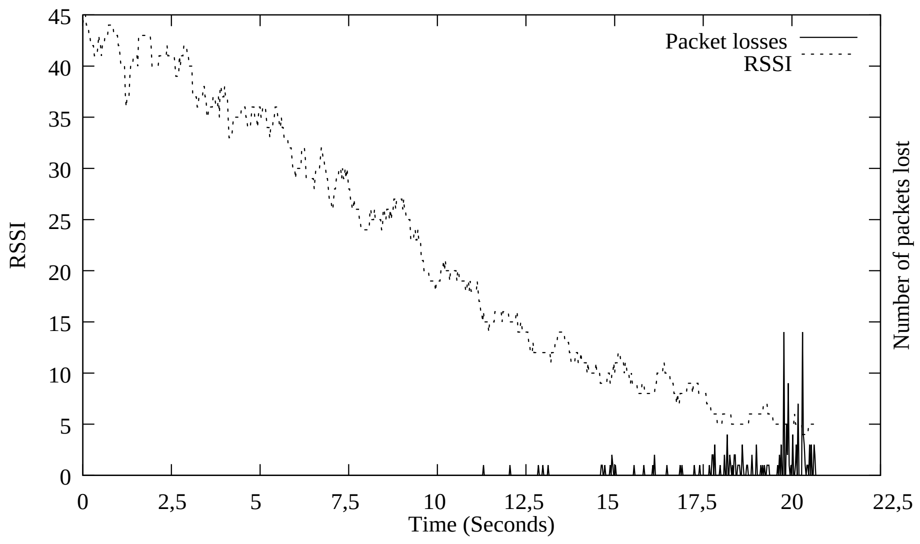

Figure 4.10: RSSI and packet losses measured for for the office corridor

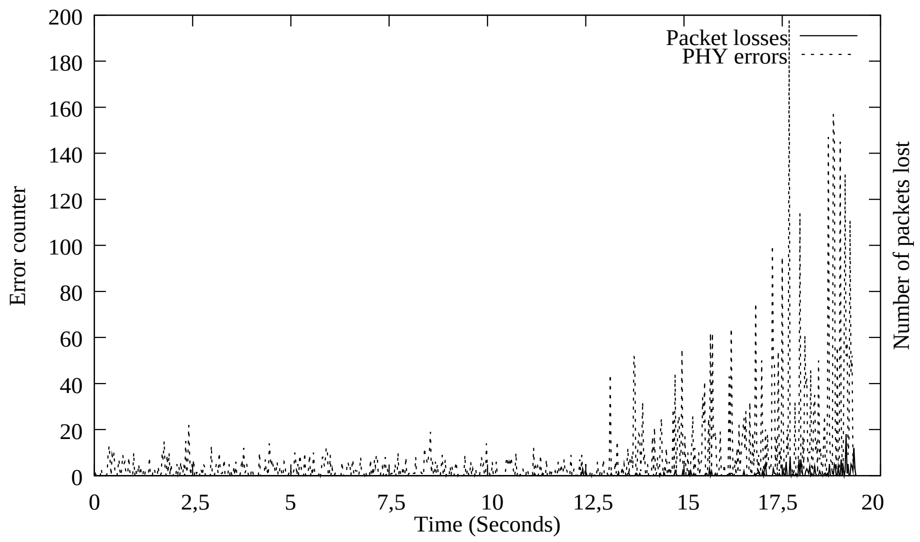

Figure 4.11: Physical errors and packet losses measured for for the office corridor

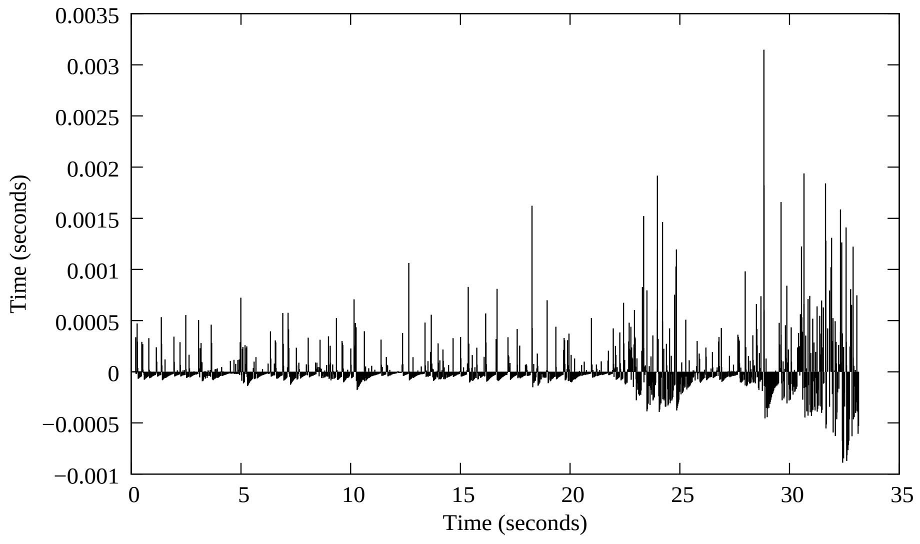

Figure 4.12: Measured jitter in the office corridor

result observed. The rise of the measured jitter indicates a deterioration in the network conditions. However the value is low and does not affect the perceived voice quality. With a sampling interval of 20 ms in Sphone, 3 milliseconds of jitter will be easily absorbed by the buffer.

Chapter 5

Handover algorithm

5.1 Description of the handover mechanism

The purpose of the previous measurement stage was to understand the behavior of these IEEE 802.11 network parameters in the measurement settings. The next task is to combine our results into a handover algorithm. In this section we propose an algorithm that uses the measured parameters and calculates a voice quality score. From this score a handover point can be determined.

5.1.1 Approach used for the trigger algorithm

The handover should be triggered by using a *handover score* that represents the current network conditions. It is worth repeating that this needs to be done *before* the voice quality degradation is audible to the user. Our task therefore is to devise an algorithm that calculates the handover score and triggers the handover sufficiently in advance of a voice quality deterioration.

We used the measurement results described in the previous section. Since different parameters have different impact, it was necessary to classify each one according to its significance. This classification will give preference to the parameters that are more sensitive and therefore react faster. The idea can be illustrated as a pyramid shown in figure 5.1. The parameters on the top provide less information, but have a closer effect for the user.

The measurements at the lower layers provide greater information about the link. However, they are not as strongly correlated with the user's perceived voice quality as parameters as higher layers. For example, the physical and link layer parameters provide information quickly, but their impact on the end user is lower compared to the effect of consecutive packet losses. The final handover score which we have selected is a combination of sub-scores weighted according to their significance.

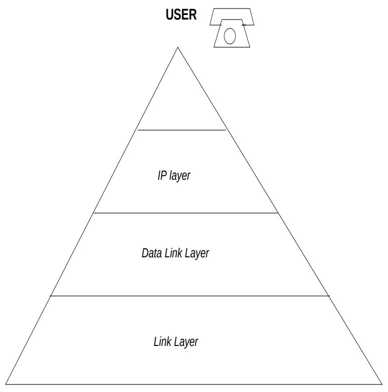

Figure 5.1: Pyramid model of the impact of network parameters

5.1.2 Timing requirements for the handover trigger

Three examples of a handover timeline are shown in figure 5.2. The horizontal line represents time. In each example the time line is divided into three periods as shown by dashed lines. The call begins on the left using WLAN. The handover is made between the points labeled t_a and t_b. At t_b the handover must be completed. The call continues to the right using the GSM network. t_b represents the instant where the perceived quality could severely degrade. According to [25] the average setup time of a GSM call is 3.6 seconds. Therefore we must have started the handover process at least 3.6 seconds before t_b to allow its completion. In our case we used a more conservative setup time of 5 seconds called T_h. In figure 5.2 (a), the point at T_h seconds before t_b is time t_a. At time t_a the handover should have been started or triggered.

5.1.3 Tradeoffs between quality and cost

[1]

It is assumed that making calls using WLAN is free, whereas GSM charges per minute of connection. In figures 5.2 (b) and 5.2 (c) we illustrate two different choices for t_a. In case 5.2 (b), the position of t_a is changed a delta time before the previous one. The total time for completion of the handover is thus increased, $T_h + \delta_h$. Using the same call setup time of case a, the handover is completed before time t_b. For a short time the call will use GSM whilst the WLAN connection could still be used, and the GSM operator will be paid for this unnecessary use. We assume that in this case the user wants to preserve the voice quality at all times, regardless of the extra cost.

In the third case, figure 5.2 (c), the opposite situation is shown. The new position of t_a is located after the original position of 5.2 (a) (shifted to the right in the graph). There is less time available for the handover process, $T_h - \delta_h$. Therefore the probability of reaching t_b before completing the handover is higher. However, if δ_h is chosen to allow the loss of only a few voice packets, call quality degradation will

[1]This section is due to the model proposed by G. Q. Maguire Jr.

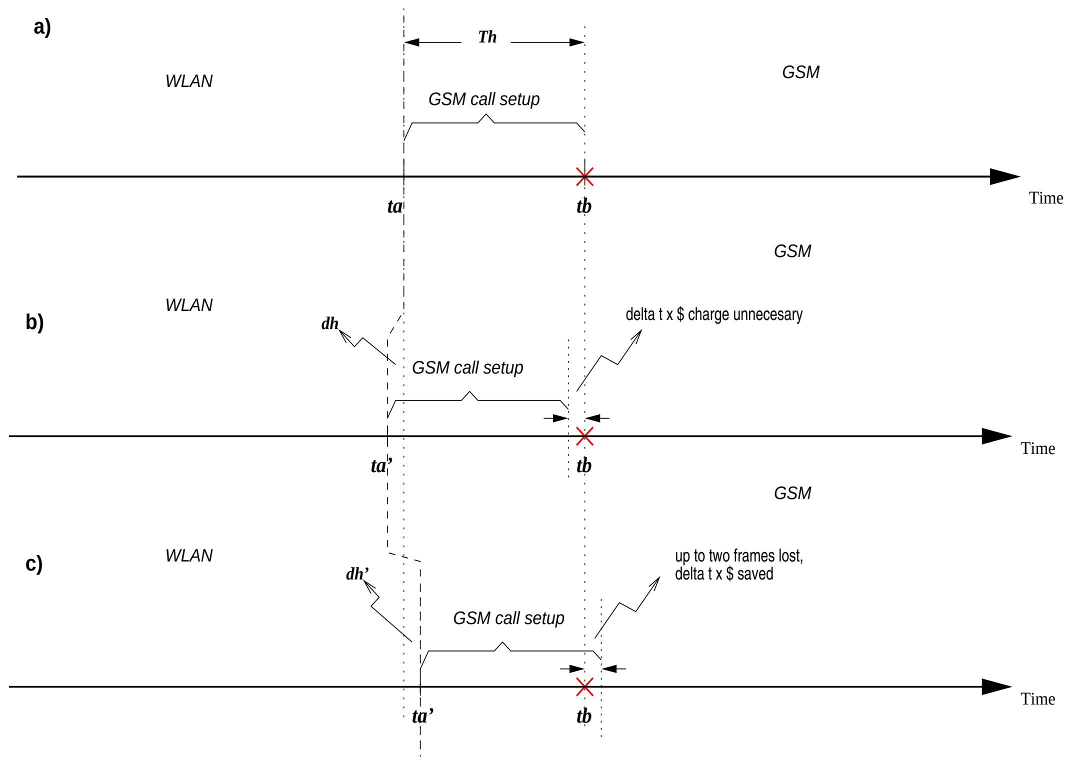

Figure 5.2: Time line for handover, with cost examples

be minimal. The total usage time of the GSM network is reduced, increasing the cost savings. In this case the user sacrifices quality for financial savings.

The algorithm that calculates the handover score should include several “cost profiles”, where the size of the window between t_a and t_b is adjustable for the user.

5.1.4 Possible handover implementations

The first implementation for the handover on a mobile phone is shown in figure 5.3(a). The dual mode mobile phone on the left is using WLAN and starts a new GSM call when the handover is triggered. When the call setup is finished the voice signals are sent via the new GSM connection.

A second choice is shown in figure 5.3(b) and 5.3(c). If this case we include GPRS as a third medium for the transmission of voice. During WLAN coverage, the mobile maintains an idle GPRS PDP context. During call setup a portion of the packets are re-routed through the GPRS link (case b). As soon as the call setup is completed the voice signal is transmitted using GSM (case c).

Assuming that the operator charges only based upon transmitted data, cost savings can be made. The packets transmitted through GPRS will maintain the voice quality if the connection to WLAN is lost. Therefore the time t_a can be shifted without further voice quality degradation.

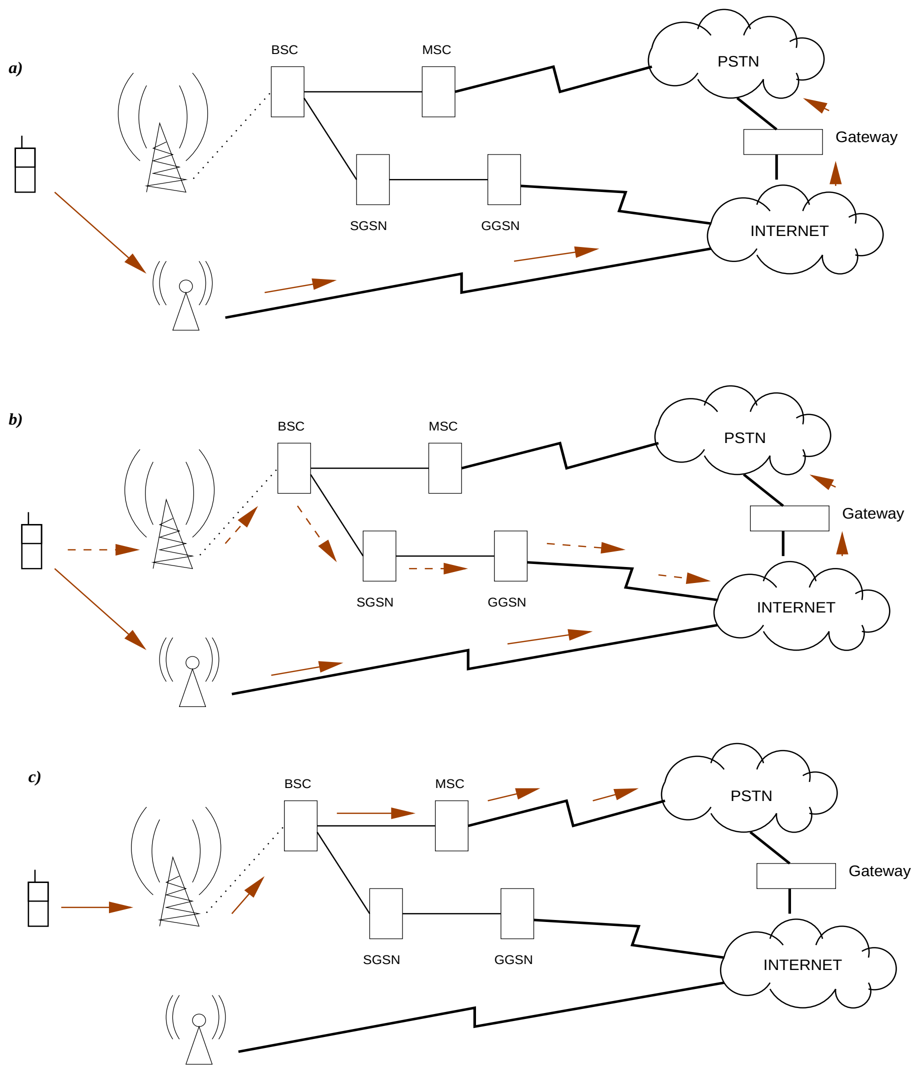

Figure 5.3: Routing of the voice data during handover. Adapted from [23]

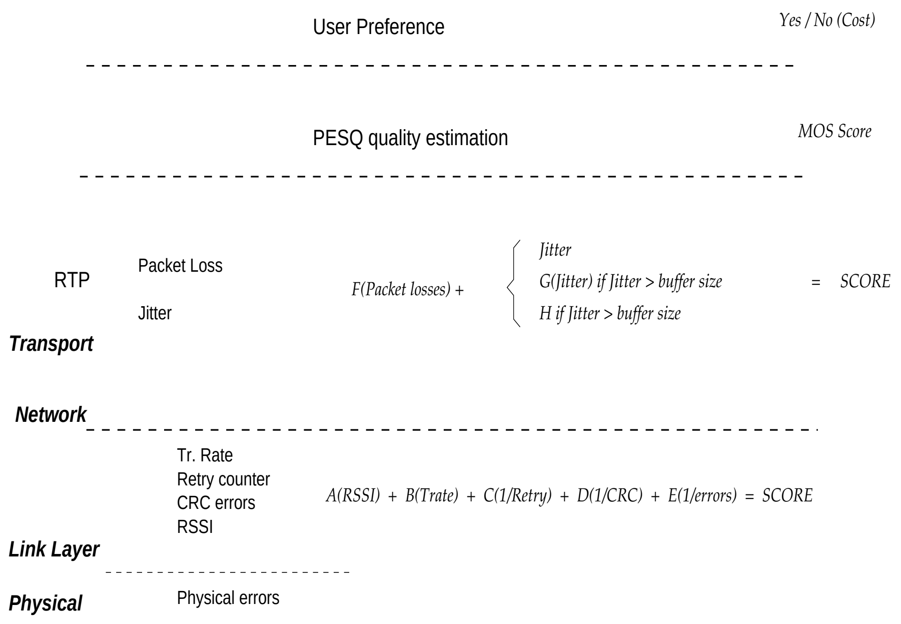

Figure 5.4: Diagram of the handover model

5.2 Structure of the trigger algorithm

We have selected parameters from the Transport (RTP), physical, and link layers. For the handover algorithm we must consider user-level information as well. Here we explain the proposed method to compute the handover score. A diagram that represents the handover algorithm is shown in the diagram 5.4.

At the physical and link layers we selected the following parameters: physical errors, CRC errors, retry counters, and RSSI. We observed that the physical, CRC errors, and retry counters increase while the signal level (RSSI) decreases (See figures 4.5, 4.6, 4.7, 4.8, 4.9). Therefore, as can be seen in figure 5.4 in order to compute the handover score we assigned inverse weights to these parameters. In this case, the sub-score will decrease if the measured error rate increases, which is the desired behavior.

At the IP layer and above we selected packet losses and jitter. Jitter influences voice quality only if its value is more than the size of the jitter buffer. Therefore, we include the jitter in the formula (i.e. as losses) only if it exceeds the buffer size. We assume that the jitter buffer size is static.

Finally a user preference selection of the handover trigger needs to be incorporated. The purpose of this is to allow the user to make a decision based on cost. The actual implementation is a simple on/off selection.

All sub-scores are then combined with additional weights. For example, if the sub-scores range from 0 to 100, a calculation of a final handover score will be computed according to the following expression:

$$\alpha * x + \beta * y + \gamma * z + \delta * v$$

Where α, β, γ, and δ are the weights assigned to user, PESQ, IP, and link layers respectively. *x,y,z* and

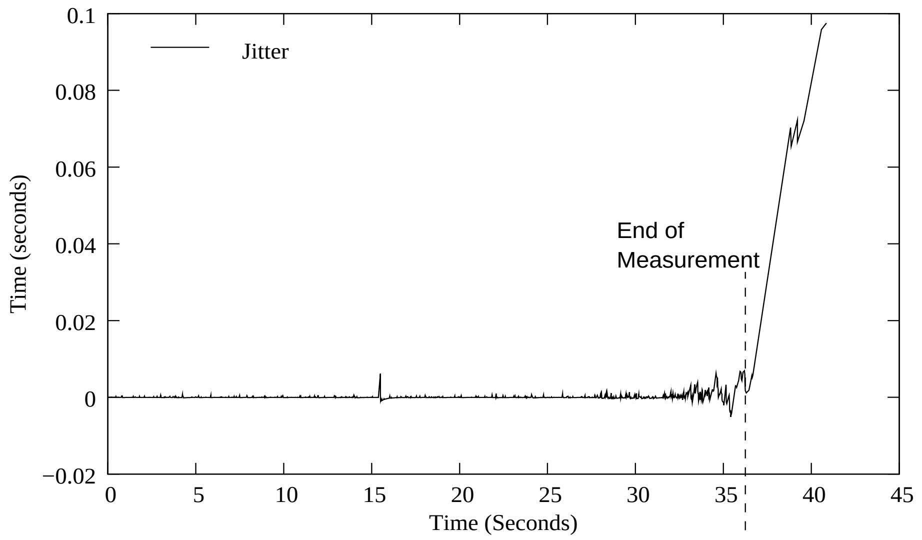

Figure 5.5: Measurement of jitter in office corridor

v are the calculated sub-scores.

5.3 Design of the handover trigger algorithm

In this section we describe the process that we used to develop the algorithm. We used the IEEE 802.11 and RTP measurements and applied statistical tools to calculate a handover score. In the first part we describe the selected tools and parameters, as well as some discarded choices. Then we explain the final design of the trigger algorithm.

5.3.1 Discarded parameters

Some parameters were not included in the final algorithm. For example, jitter was only considered when it exceeded a value of 20 milliseconds, mainly because of the de-jitter buffering process that neutralizes its effect. However, in practice the jitter never reaches this value until the end of the measurement, as it can be seen in figure 5.5.

There were other discarded parameters, such as short retry counters and transmitted management frames. Their measurements showed little correlation to changes in network conditions. A special case was the transmission rate. This parameter was considered at the beginning, but measurements showed an unexpected result - which is that it did not correlate well with future network conditions. Hence, it was discarded from the handover algorithm. However this parameter might be considered in future work.

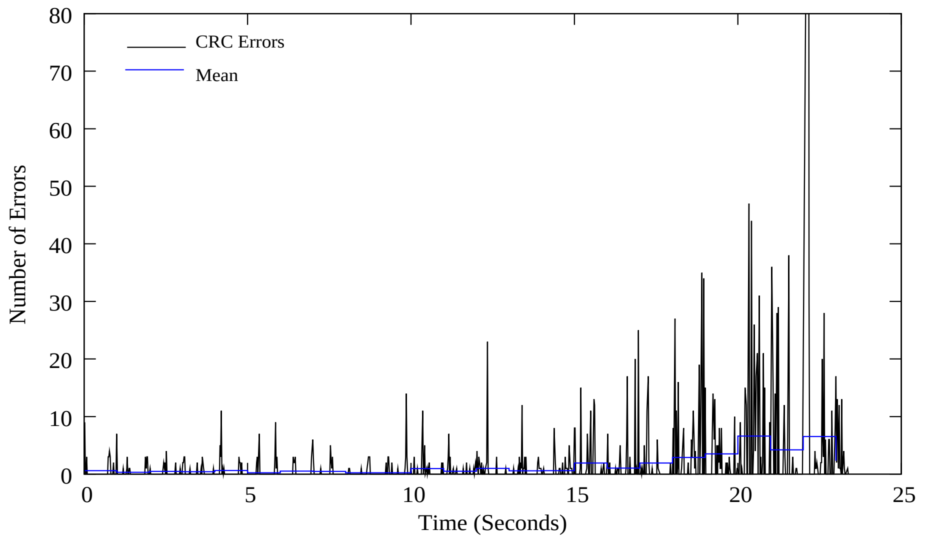

Figure 5.6: CRC Error counter in the office corridor with the mean

5.3.2 The use of statistic tools

In general, the results of all the parameters show rapid changes over small periods of time. An increasing or decreasing trend can be easily detected by visual inspection of the plots. However, it would be difficult to implement a smooth and decreasing handover score with those values. We have tested different statistical tools that work as a ”low pass” filter on the measured values. This analysis and our final choices are described below.

5.3.3 Mean and variance

We calculated the mean as the sum of the measured values divided by the number of observations, over a fixed amount of time. The mean expresses the average of all values in this observation period. The variance is used to calculate the amount of variation of the parameter in relation to the mean. For our analysis we used the standard deviation, i.e., the square root of the variance. The standard deviation is expressed in the same units as the original parameter. This was necessary in order to analyze and compare our graphs.

In our measurements, the IEEE 802.11 and RTP parameters are constantly changing. However, an automatic handover calculation requires a stable increment or decrement of the value. We utilized the mean and standard deviation to provide stable results over fixed periods. They were calculated for all link layer and RTP parameters over 1-second observation intervals. The results for the CRC, Physical error counters, and packet losses are shown in figures 5.6, 5.7, and 5.8.

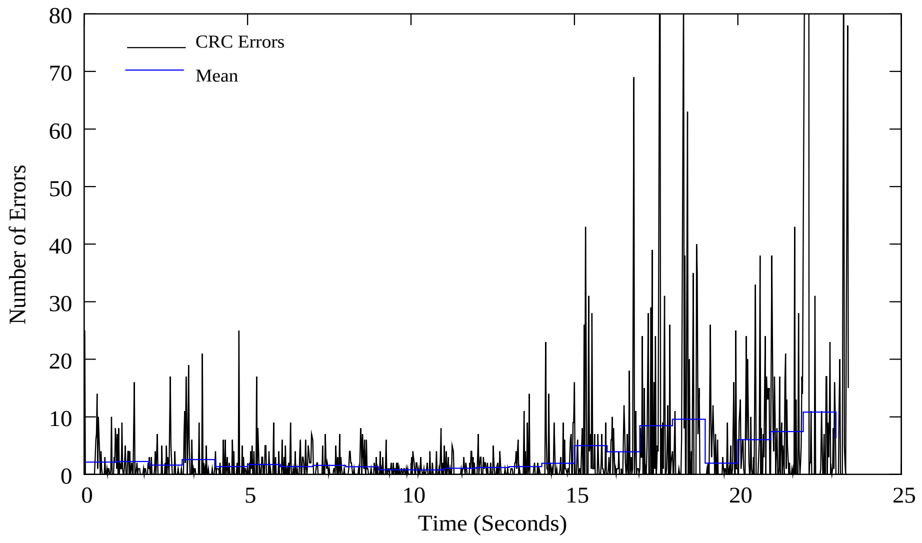

Figure 5.7: Physical error counter in the office corridor with the mean

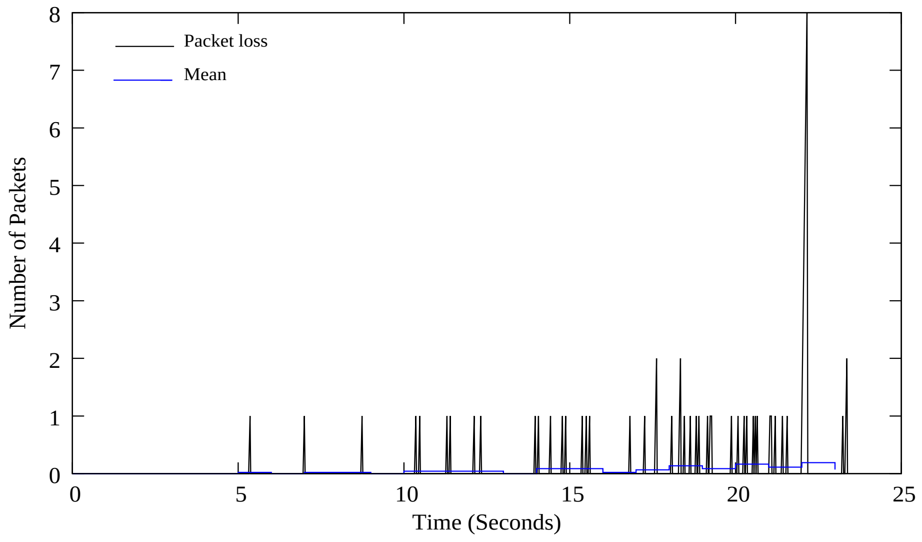

Figure 5.8: Packet losses in the office corridor with the mean

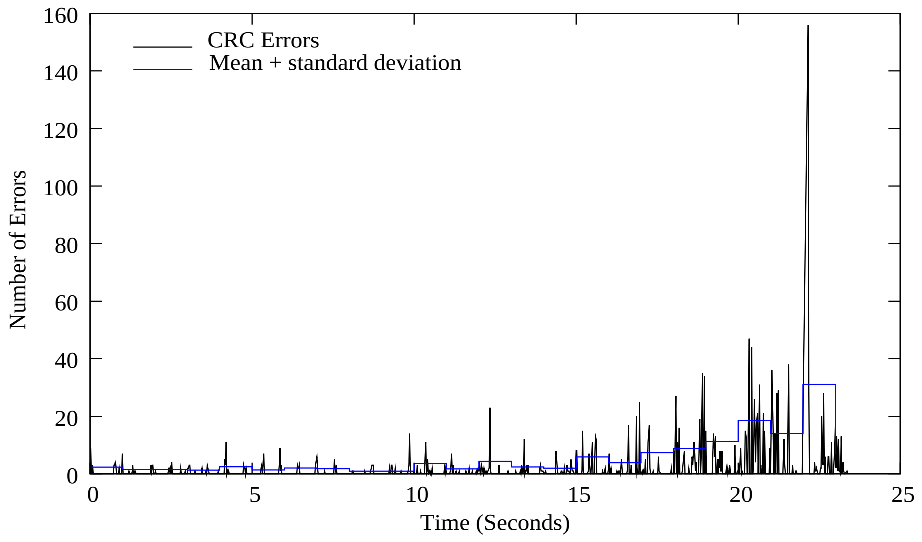

Figure 5.9: CRC Error counter in the office corridor with mean added to standard deviation

5.3.4 Standard deviation and counting of events

As shown in the previous section, the mean provides a stable result. However since the value of most of the IEEE 802.11 parameters is zero during a long part of the measurement, the mean is small and reacts slowly with a rapid increase in the values. In order to better track these changes, we first combined the standard deviation and the mean. We added the mean and the standard deviation for the same observation interval of 1 second. The result is shown figures 5.9, 5.10, and 5.11.

A second calculation counts the number of events recorded in a period and multiples them by the mean. As seen in figure 5.12 this calculation tracks more closely the actual value of the packet losses. In CRC and physical error counters we observed excessive increases of the resulting value. This will affect the implementation of the trigger, therefore we discarded the use of this method.

5.3.5 Analysis of exponential moving averages

A third method used is called Exponential Weighted Moving Averages (EWMA). An analysis of moving averages is used for smoothing a time-series of data. The calculation is called “simple” when the arithmetic average is constantly done over a fixed period before the point of observation. In our case we chose the “exponential” method, which gives a multiplier with greater weight given to more recent values. The EWMA is calculated according to the following formula [3]:

$$\mathbf{E}(Current) = ((\mathbf{A}(Current) - \mathbf{E}(previous))\mathbf{multiplier})) + \mathbf{E}(previous)$$

Where:

$$\mathbf{multiplier} = \frac{2}{\mathbf{periods}+1}$$

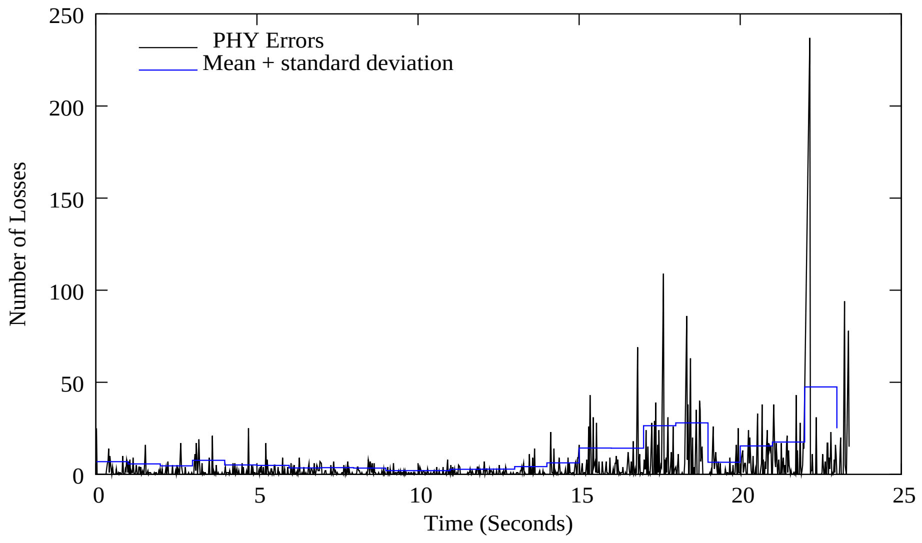

Figure 5.10: Physical error counter in the office corridor with mean added to standard deviation

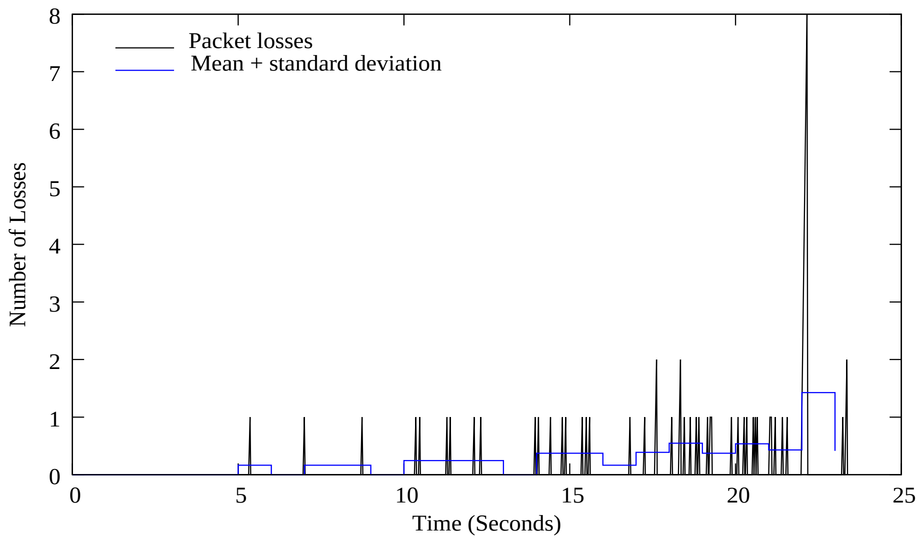

Figure 5.11: Packet losses in the office corridor with the mean

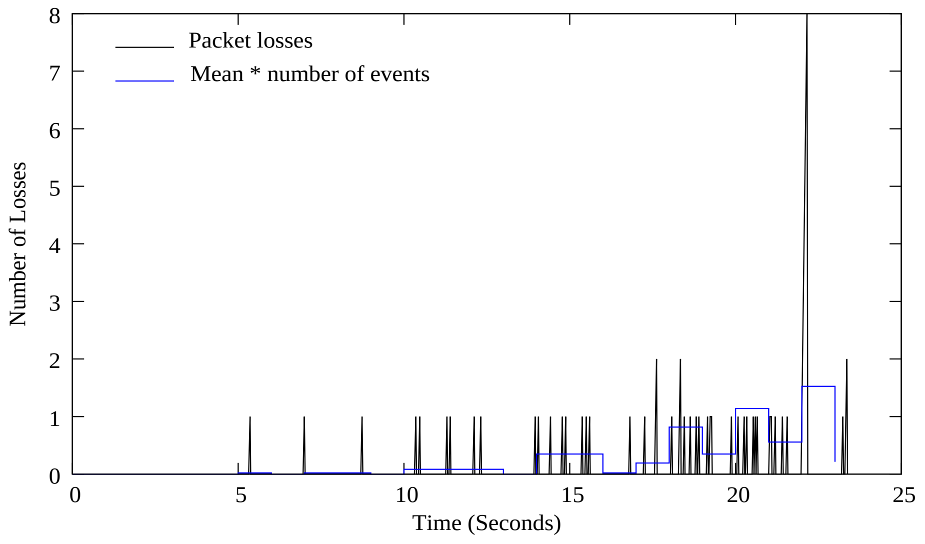

Figure 5.12: Packet losses in the office corridor with a calculation of the mean times the number of events

E is the result of the exponential moving average and A is the vector of data. The multiplier provides the weight. It is calculated from the number of measurements (periods) considered for the average.

This test proved useful for smoothing the RSSI, and the result of the exponential moving average is shown in figure 5.13.

5.3.6 Selection of the smoothing method, weighting, and final algorithm

To achieve the goal of a smooth handover function, we tested two different methods for the calculation of a handover trigger. We first tried the calculations based on the mean as described above, and observed in the results. We selected the method of adding the standard deviation to the mean. It was used to smooth out the plots of the long retry, CRC counter, and physical error counters. The second method, counting of events multiplied by the mean, was selected for the RTP parameters: packet loss and jitter. The jitter parameter was only considered when it exceeds a 20-millisecond threshold. We used an observation interval of one second over which to calculate the mean.

The following step was to normalize each value to fit a range from 0 to 100. The results were six individual scores from 0 to 100. The resulting values for each parameter were arithmetically added. We experimented with different sets of weights for the parameters. By observing the resulting function we determined that a set of equal weights was most appropriate.

We needed a smooth function in order to better detect the trend of decreasing network conditions. After several experiments we only obtained a flat function using equal weights for all parameters. If we give greater preference to one over the others the fluctuation increases significantly. Therefore we kept equal weights in the final function. The handover score obtained using this method is shown in figure 5.14.

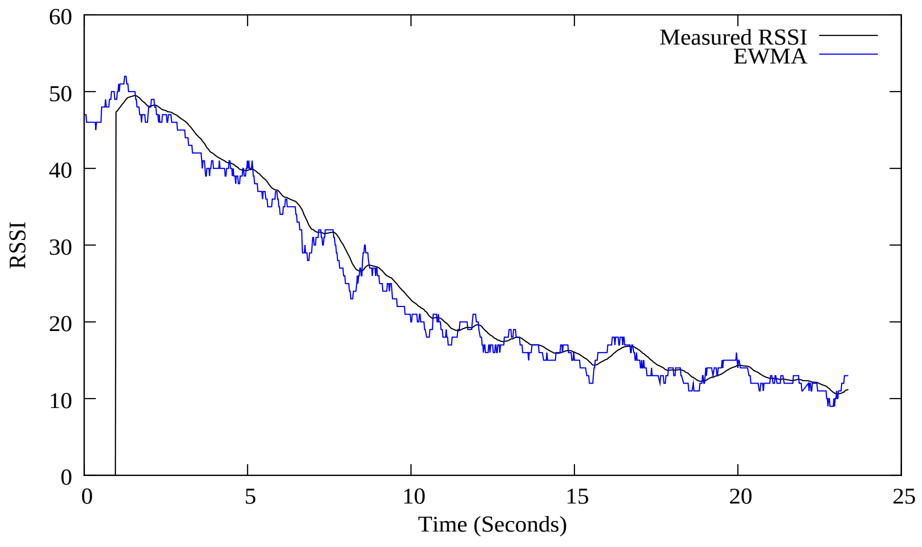

Figure 5.13: Result of EWMA calculation on RSSI, for the corridor test

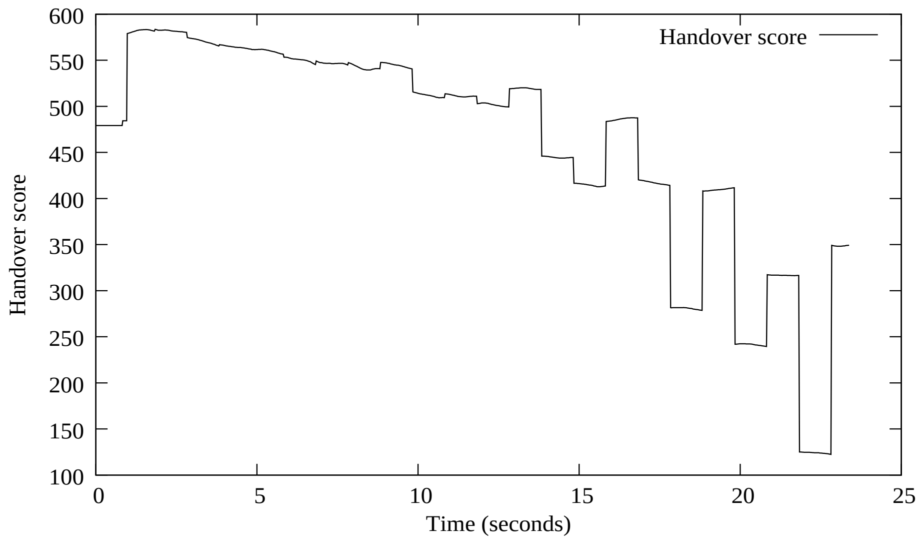

Figure 5.14: Handover score calculated from the mean and standard deviation

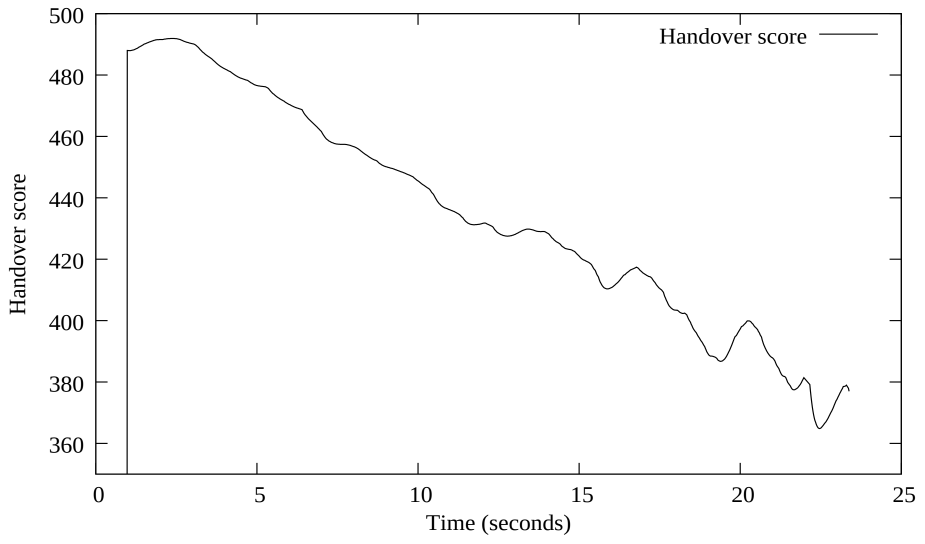

Figure 5.15: Handover score calculated using exponential moving averages, 2% weight

We evaluated the resulting function. In figure 5.14, the function jumps more than 50 points from second 14 to 15. This is caused by the accumulating effect when all the parameters increase. The effect on the handover score function is a sudden decrease. If a lower mean is measured in second 15, the handover score returns to their original value, 50 points higher. This may lead to confusion in determining the exact point when to trigger the handover. If the conditions of the network recover, the handover will be triggered unnecessarily or too early. Therefore this method was discarded.

In a second experiment we used only the EWMA calculations for all the parameters instead of the mean. The results of each parameter were combined as explained in section 5.3.5. The resulting score was also smoothed using EWMA.

We tried two different multipliers and observed how smooth the resulting plot was. Figures 5.16 and 5.15 show the result with a constant equivalent to 2% and 4% weight on the current measurement respectively. The calculation using 4% produces a more stable output. We selected this value for the final handover score function.

We estimated that a handover trigger should occur five seconds before a deterioration on the voice quality. Based on our measurements, we estimated that the handover should be triggered when our score reaches 420. This corresponds to a reduction of 16% from the maximum value. We chose this threshold for the algorithm evaluation.

5.4 Performance evaluation of the handover trigger

It was necessary to evaluate whether the implemented function was effective or not. Our algorithm should be able to trigger a handover within 5 seconds prior to poor voice quality. Based on the previous measurements,

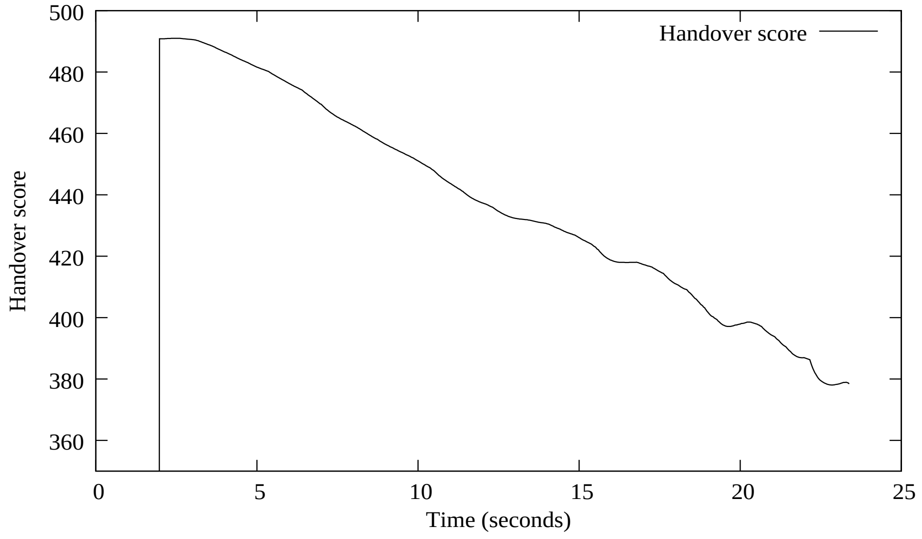

Figure 5.16: Handover score calculated using exponential moving averages, 4% weight

we designed an experiment to compare the handover trigger algorithm with the quality of the recorded sound file. We used the same office corridor, tools and configuration as described in section 3.4.2.

We selected two routes with different signal conditions, shown in figure 5.17. The first route is shown as a solid red line and was the same used for all previous experiments. The second route is marked in dashed green and covers the same distance along another corridor. The wall in between causes a greater attenuation of the signal on the second route.

By listening to the recorded audio, in the first route the perceived voice quality does not deteriorate to a noticeable level. Therefore for this route a handover would not be necessary. In the second case the received audio is severely degraded. We assume that in the second route a handover should be triggered. Two speech samples of 1 minute were recorded for this experiment. For each one a male voice reads a magazine article in English.

In figure 5.18 we show the resulting handover score for the experiment while walking along the second route. The receiving laptop was moved along the corridor until the transmission was lost. We listened to the prerecorded speech sample and evaluated the subjective voice quality. After 31 seconds the number of glitches and losses indicated a significant deterioration of network conditions, as marked by the right blue dashed line. The point where the handover should have been triggered, 5 seconds before second 31, is shown with the left dashed line in figure 5.18. At this point the handover score shows a value of 415 approximately. In figure 5.19 a second measurement is shown. This time the deterioration in voice quality is observed at 31 seconds. The value of the handover score in the second case is approximately 425. In this example our algorithm successfully triggers the handover.

The first route is an example of a case when the voice quality experiences little deterioration, as the

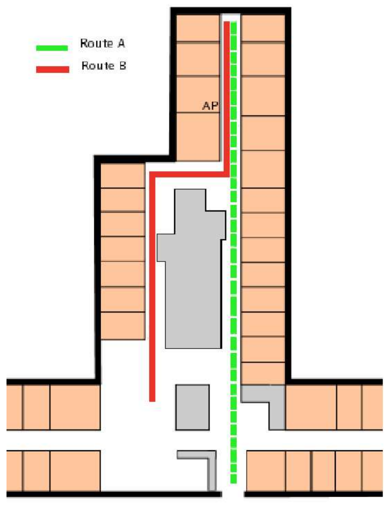

Figure 5.17: Plan of office corridor with a description of routes for evaluation of the handover

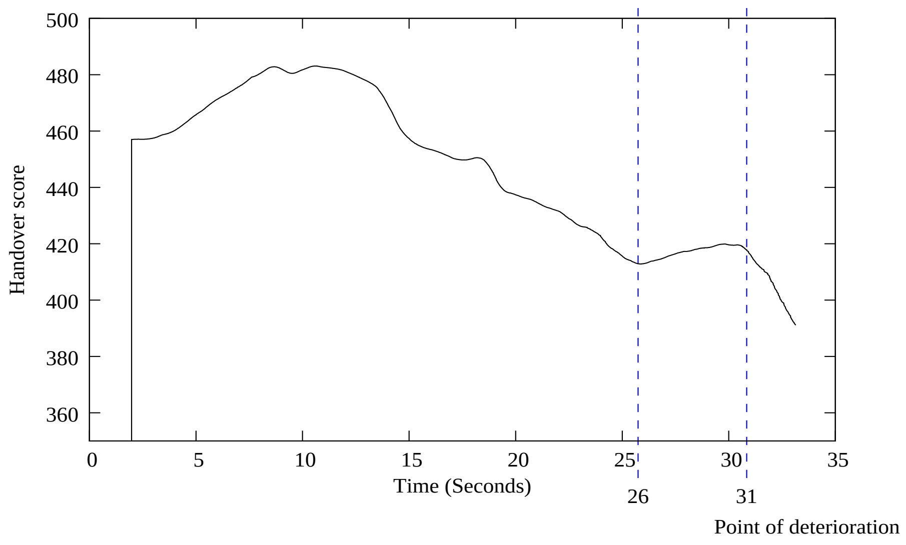

Figure 5.18: Plot of handover score for the experiment in the second route, the points of deterioration are marked in dashed blue

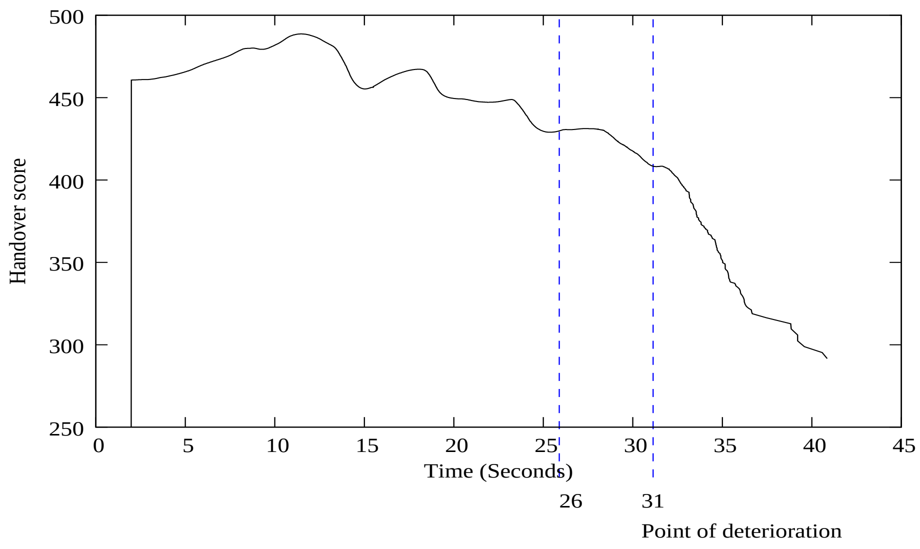

Figure 5.19: Plot of handover score for a second experiment in the second route, the points of deterioration are marked in dashed blue

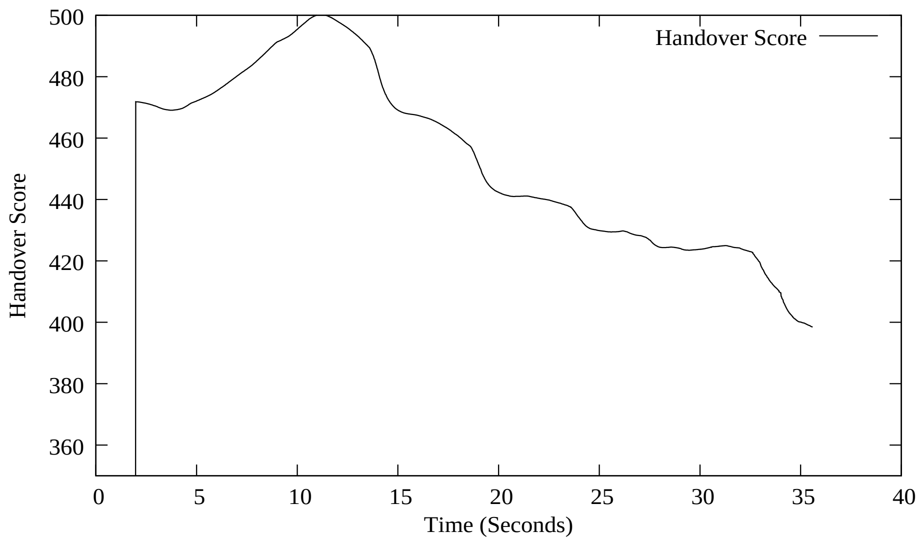

Figure 5.20: Plot of handover score for a second experiment along the first route

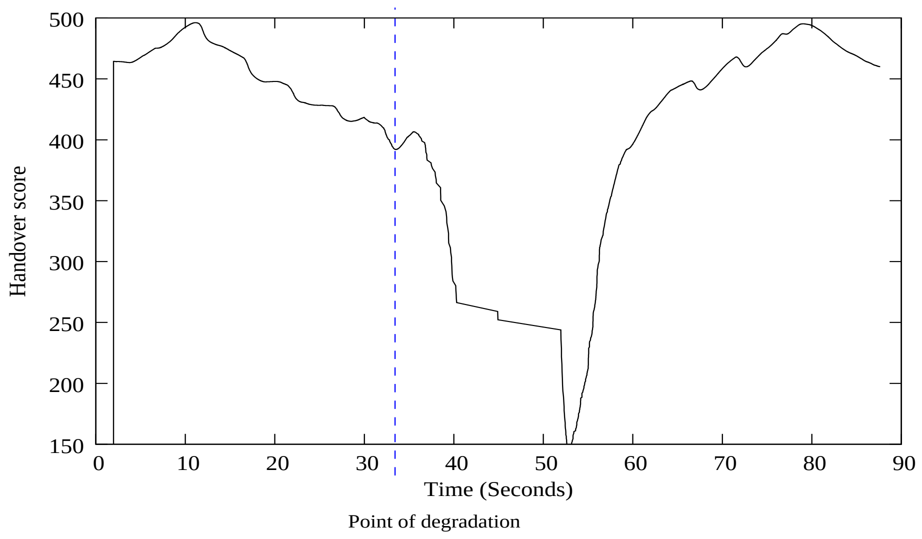

Figure 5.21: Plot of handover score for an experiment with return walk using the second route

AP is always in line of sight and the recorded sound has no interruptions. We repeated the experiment on this route. The resulting handover score is shown in figure 5.20. In this case the plot shows a more stable function.

The score is 420 at the end of the measurement. According to the algorithm, a handover should occurred at this point. However, the recorded sound indicated no deterioration. Therefore an unnecessary trigger would have been activated in this case. In conclusion, the algorithm works only when the user consistently moves further away from the Access Point. If a person reaches the boundaries of IEEE 802.11 coverage but then he returns, the algorithm fails.

5.4.1 Example walking both ways

An experiment where a user walks away, turns and returns to the initial point was performed. We wanted to evaluate our handover score function with a user walking in both directions. We used the second route of figure 5.17 waling until the end of path indicated by the green dashed line, then we turned around and walked back to the beginning of the corridor. The same experimental configurations were used. The plot of the resulting handover score function is shown in figure 5.21.

We expected the algorithm to return to its original value. The handover score decreases until the moment when the signal is lost. According to the subjective evaluation by a listener the average point to significant degradation of voice quality is at 34 seconds. This is shown by a blue dashed line. After this point the handover score in the plot decreases due to disconnection, five seconds before the degradation point the handover value is 420 points. In conclusion, for this case a handover could have been triggered successfully. In the case of the return trip it was not possible to accurately evaluate the recorded sound, because of a few seconds of lost connection that were not recorded. On the file the sound is recovered immediately after the connection is lost. There was a gap of silence that was not included on the file and it disturbed the timing information in the recorded sample. However, by visual inspection of the plot in figure 5.21, it can be seen that the function recovers its value. The handover score function also seems to work in the opposite direction of change.

5.5 Discussion

In this section we discuss the alternatives that were considered during the design of the handover algorithm. Other discarded options are mentioned as well.

We considered the use the transmission rate as a parameter for the handover trigger. The transmission rate is adapted by the hardware according to the network conditions. However, it was discarded because the provided information was not useful. We also considered including other parameters such as the signal level and noise. The signal level was not used because it provides the same information as the RSSI. In addition, the Netgear card and the Madwifi driver did not measure noise accurately.

Once the parameters were selected, we observed high fluctuations of the resulting handover functions. We tried two different methods to address this issue and filter rapid changes of the parameter. The first method, based on the mean and standard deviation, was discarded. We selected the second method: using exponential weighted moving averages.

These calculations of mean and standard deviation required a fixed observation interval. We selected a value of one second which offered relative stability, and the ability to track the original value in a five-second call setup time. However, we still observed substantial drops and changes in the handover score in the boundaries between observation periods. This can be seen between seconds 13 and 14 in figure 5.14. It was difficult to define a threshold for the handover trigger using this method. In addition, using a one-second observation time, the value at any point can only be obtained once every second. Therefore we discarded this method and decided to calculate the trigger using EWMA.

In the case of the weighting used for the EWMA analysis, using a weight of 2% the fluctuation of the parameter is too high. However, using 2% weight the result better tracks the actual value. We chose to use 4% weight which is less accurate, but offers less fluctuation.

Another discarded tool was the use of RTCP information. The implementation of RTCP in Sphone reported values less frequently than our direct measurement. It sent RTCP reports every five seconds. However, this method can be included in future work.

Based on the results of our evaluation measurements, it is possible that the algorithm does not trigger the handover on time or unnecessarily in some special cases. We determined that in the first case of a sudden deterioration of the signal it works as expected. A second case which works is when the user walks up to the limits of coverage of the AP and instead of going further away he comes back. Other evaluation tests were not included due to limitations of time and resources. Transmitting traffic in both directions and the use of IEEE 802.11g are experiments that are left for future work.

5.6 Conclusions

In this masters thesis we measured IEEE 802.11 and RTP network parameters and implemented a handover trigger. We designed an experiment based on a typical handover situation to measure a set of common IEEE 802.11 and RTP parameters. We analyzed the experimental results by comparing the resulting plots. In this part we selected six parameters that were useful for extracting information about network conditions.

In the second part we presented a method to make a handover decision during a VoIP call using a WLAN to make a vertical handover to a cellular system. Our goal was to design a handover algorithm based on multiple measurements rather than only traditional signal-to-noise information. We implemented a prototype of such an algorithm based on the selected parameters.

This algorithm was designed to be implemented completely in the mobile host, and to be totally independent of the cellular operator. Our implementation was based on a modification of open source software tools for IEEE 802.11 and RTP. The software was tested only on a laptop computer and not on a real mobile platform.

We consider that the method of measuring IEEE 802.11 and RTP parameters can be applied in real networks. The information they provide is easy to measure, accurate, and with high granularity. No major changes in the protocol nor in the hardware are required.

The algorithm was evaluated in typical office scenarios. We measured the performance of the algorithm under two different handover situations. We determined that for an average situation the handover score algorithm can be used to successfully trigger a handover. However, the design is not flawless. In some situations the handover would be triggered too early or unnecessarily. In a future implementation these issues will have to be addressed.

5.7 Future work

Several experiments can be suggested for future work. Due to limitations of time and resources it was not possible to evaluate duplex traffic. Further modifications to the Sphone can be implemented to achieve this goal. It should be possible to test different implementations as well. Uplink traffic can be analyzed and all possible effects on the handover score algorithm can be evaluated.

We only implemented this algorithm in a simulation of the environment. Moreover, this model of the environment was limited to a single office and a limited set of experiments. The measurements can be enhanced with tests include in other environments: home, outdoor, or different office buildings. It should be possible to use a larger IEEE 802.11 network. Multiple Access Points can be used as well to observe the behavior of the algorithm. The trigger method can be tested for different types of handover. Handovers between IEEE 802.11 Access Points and between IEEE 802.11 and GSM can be included into the implementation.

Future work may include first an integration with a real handover procedure, by implementing a handover method and testing the trigger. Secondly, we need to evaluate the feasibility to have it implemented in a mobile phone.

Finally, the handover trigger algorithm can be improved. It should be possible to include solutions for the situations where the present code fails. It might be able to detect when the user turns back when the signal recovers. The algorithm should be able to cancel a handover trigger and avoid unnecessary GSM charges.

Appendix A

Code Listings

Modifications to file athstats.c, Madwifi version 0.9.2

```
/* New declarations */
/* libraries required for timestamps */
#include <time.h>
#include <sys/timeb.h>

/* structure to store the parameters */
struct wlandata {
  u_long inputf;
  u_long outputf;
  uint32_t lretry;
  uint32_t crcerr;
  uint32_t phyerr;
  uint32_t athrssi;
  uint32_t rtpseq;
  uint32_t rtptime;
  uint32_t rtpfrac;
};

/* end of declarations */

/* Added code to main (renamed main_ath to be called by sphone) */
/* line 240 */
/* variable defined in time.h, long integer used to store */
/* EPOCH seconds with function time() */
time_t t1;
/* used to store miliseconds value */
short i;
/* struct defined in timeb.h, contains fields of time, miliseconds, timezone */
struct timeb tp;

/* creates struct of statistics defined in ath_impl.h */
struct ath_stats total;
u_long itot, otot;

 /* routine to get time and milliseconds from local time*/
(void) ftime(&tp);
```

```
i = tp.millitm;
t1 = tp.time;

/* line 305 */

/* store data in structure */

wdata->inputf = itot - total.ast_rx_mgt;
wdata->outputf = otot;
wdata->lretry = total.ast_tx_longretry;
wdata->crcerr = total.ast_rx_crcerr;
wdata->phyerr = total.ast_rx_phyerr;
wdata->athrssi = total.ast_rx_rssi;
wdata->rtpseq = secnum;
wdata->rtptime = tstsec;
wdata->rtpfrac = tstfrac;
```

Modifications to file iwconfig.c, Wireless Tools v.28

```
/* New declarations */

#include "iwlib2.c" /* libraries to handle timestamps */
#include <time.h>
#include <sys/timeb.h>

struct wconfig {
  int trate;
  int tpower;
  int sec;
  int ms;
};

/* line 306 */

pconfig->trate = atoi(buffer);

/* line 541 */

time_t t1;
short i;
/* struct defined in timeb.h, contains fields of time, miliseconds, timezone */
struct timeb tp;

  rc = get_info(skfd, ifname, &info);

(void) ftime(&tp);
i = tp.millitm;
t1 = tp.time;
```

Modifications to file iwconfig.c, Wireless Tools v.28

```
/* New declarations */

#include "iwlib2.c" /* libraries to handle timestamps */
```

```
#include <time.h>
#include <sys/timeb.h>

struct wconfig {
  int trate;
  int tpower;
  int sec;
  int ms;
};

/* line 306 */

pconfig->trate = atoi(buffer);

/* line 541 */
time_t t1;
short i;
/* struct defined in timeb.h, contains fields of time, miliseconds, timezone */
struct timeb tp;

(void) ftime(&tp);
i = tp.millitm;
t1 = tp.time;

/* line 1438 */
/* changed name to main, declared structures config mconfig, */
/* modified call to print_info to include struct */

 print_info(skfd, "ath0", NULL, 0, &config);
  mconfig->trate = config.trate;
  mconfig->tpower = config.tpower;
  mconfig->sec = config.sec;
  mconfig->ms = config.ms;
  /* Close the socket. */
  iw_sockets_close(skfd);
```

Modifications to file sphone_rcv.c, Wireless Tools v.28

```
/* Added declarations by Daniel */

struct wlandata {
  u_long inputf;
  u_long outputf;
  uint32_t lretry;
  uint32_t crcerr;
  uint32_t phyerr;
  uint32_t athrssi;
  uint32_t rtpseq;
  uint32_t rtptime;
  uint32_t rtpfrac;
};

struct wconfig {
  int trate;
  int tpower;
  int sec;
  int ms;
};

/* line 223 */

/* Declarations added in receive_and_playout function */
/* libraries required for timestamps */
#include <time.h>
#include <sys/timeb.h>

/* structure to store the parameters */
struct wlandata {
  u_long inputf;
  u_long outputf;
  uint32_t lretry;
  uint32_t crcerr;
  uint32_t phyerr;
  uint32_t athrssi;
  uint32_t rtpseq;
  uint32_t rtptime;
  uint32_t rtpfrac;
};

/* end of declarations */

/* Added code to main (renamed main_ath to be called by sphone) */

/* line 240 */

/* variable defined in time.h, long integer used to store */
/* EPOCH seconds with function time() */
time_t t1;
/* used to store miliseconds value */
short i;
/* struct defined in timeb.h, contains fields of time, miliseconds, timezone */
struct timeb tp;

/* creates struct of statistics defined in ath_impl.h */
struct ath_stats total;
u_long itot, otot;

 /* routine to get time and milliseconds from local time*/
(void) ftime(&tp);
i = tp.millitm;
t1 = tp.time;

/* line 305 */

/* store data in structure */

wdata->inputf = itot - total.ast_rx_mgt;
wdata->outputf = otot;
wdata->lretry = total.ast_tx_longretry;
wdata->crcerr = total.ast_rx_crcerr;
wdata->phyerr = total.ast_rx_phyerr;
wdata->athrssi = total.ast_rx_rssi;
wdata->rtpseq = secnum;
```

```
wdata->rtptime = tstsec;
wdata->rtpfrac = tstfrac;

   struct wlandata ath_current;
   static struct wlandata ath_total;
   struct wconfig iwconf_current;
   static uint32_t initial_rtptime;
   static double initial_rtpseconds;
   static double initial_nowtime;
   unsigned int formatted_rtptime;
   unsigned int formatted_nowtime;
   double temp_rtpsec;
   double temp_nowsec;
   static double deltatime;
   static double adjtime;
   static double packet_jitter;
   int hScore;
   int tempPHY;
   int tempCRC;
   int tempRETRY;
   int tempLoss;
   struct timeval rtpTimeval;

/* line 304 */

/* call added by Daniel */

    NTP2TV(&hdr.ts, &rtpTimeval);

        main_ath(hdr.seq, rtpTimeval.tv_sec, rtpTimeval.tv_usec, &ath_current);
main_iwconfig(&iwconf_current);
FILE *Fp;
Fp = fopen("/home/daniel/athdata.txt", "a");

temp_rtpsec = timeval2sec(rtpTimeval);
temp_nowsec = timeval2sec(now);

if (ath_current.rtpseq == 0){
  initial_rtptime = ath_current.rtptime;
  initial_rtpseconds = temp_rtpsec;
  initial_nowtime = temp_nowsec;
  deltatime = 0;
  packet_jitter = 0;
    }

if (ath_current.rtpseq == 1){
  packet_jitter = 0;
}


adjtime =  calcjitter(temp_nowsec, temp_rtpsec, &deltatime,\\
        &packet_jitter);

tempPHY = ath_current.phyerr - ath_total.phyerr;
tempCRC = ath_current.crcerr - ath_total.crcerr;
tempRETRY = ath_current.lretry - ath_total.lretry;
tempLoss = ath_current.rtpseq - ath_total.rtpseq - 1;
```

```
hScore = handoverScore(tempPHY, tempCRC, tempRETRY, ath_current, \\
        iwconf_current, tempLoss, packet_jitter);

fprintf(Fp, "%8u %8u %8u %8u %d %8u %8u %F %F %F %F %F %F\n",
ath_current.lretry - ath_total.lretry,
ath_current.crcerr - ath_total.crcerr,
ath_current.phyerr - ath_total.phyerr,
ath_current.athrssi,
iwconf_current.trate,
ath_current.rtpseq - ath_total.rtpseq - 1,
ath_current.rtptime - initial_rtptime,
packet_jitter,
temp_rtpsec - initial_rtpseconds,
temp_nowsec - initial_nowtime,
temp_rtpsec,
temp_nowsec,
deltatime
);
fclose(Fp);

ath_total.inputf = ath_current.inputf;
ath_total.outputf = ath_current.outputf;
ath_total.lretry = ath_current.lretry;
ath_total.crcerr = ath_current.crcerr;
ath_total.phyerr = ath_current.phyerr;
ath_total.athrssi = ath_current.athrssi;
ath_total.rtpseq = ath_current.rtpseq;
   ath_total.rtptime = ath_current.rtptime;
ath_total.rtpfrac = ath_current.rtpfrac;
```

New file octavehandover.m, for load in octave-matlab. *A* is the matrix of results obtained from *stdout* after executing the test.

```
/* Declare variables */
l = length(A);
t = A(l,7);
D = zeros(1,(t + 1));
x = 0;
i = 1;

/* Calculate number of entries within first second */
while (A(i,7) == 0)
  x++;
  i++;
endwhile

D(1,1)= x;
x = 0;

  for i = (1 : t)
    for j = (1 : l)
      if (A(j,7) == i)
x++;
      endif
    endfor
D(1,(i + 1)) = x;
```

```
    x = 0;
  endfor

/* Declare auxiliary matrix for calculations on first second*/
M = zeros((t+1),13);
V= zeros((t+1),13);
S = zeros((t+1),13);
N = zeros((t+1),13);
O = zeros((t+1),13);
C = zeros((t+1),13);

/* sequence that goes through all the parameters */
for k = (1 : 6)

   n = D(1,1);
   T = zeros(n,1);

   i = 1;

/* calculate statistics for first second */
   while (A(i,7) == 0)
     T(i,1) = A (i,k);
     i++;
   endwhile

   M(1,k) = mean(T);
   V(1,k) = var(T);
   S(1,k) = std(T);
   N(1,k) = M(1,k)+S(1,k);
   O(1,k) = M(1,k)-S(1,k);
   C(1,k) = sum(T)*M(1,k);

/* calculate statistics for following seconds until the end */
   for i = (1 : t)
     n = D(1,i);
     T = zeros(n,1);
     z = 1;
     for j = (1 : l)
       if (A(j,7) == i)
 T(z,1) = A (j,k);
         z++;
       endif
     endfor
     if (D(1,i) > 0)
    M((i + 1),k) = mean(T);
    V((i + 1),k) = var(T);
    S((i + 1),k) = std(T);
    N((i + 1),k) = (M((i + 1),k)+S((i+1),k));
    O((i + 1),k) = (M((i + 1),k)-S((i+1),k));
    C((i + 1),k) = (sum(T)*M((i + 1),k));
     endif
   endfor
endfor
```

Bibliography

[1] Kismet: 802.11 layer2 wireless network detector, sniffer, and intrusion detection system. http://www.kismetwireless.net/. Accessed 29th August 2006.

[2] Madwifi: Multiband atheros driver for wifi. http://madwifi.org/. Accessed 26th August 2006.

[3] Moving averages, introduction. http://stockcharts.com/. Accessed 20th October 2006.

[4] Optimobile AB. Optimobile homepage. http://www.optimobile.se/. Accessed 22th July 2006.

[5] S.M. Ali. VoWiFi Roaming. Master's thesis, KTH Royal Institute of Technology. Stockholm, Sweden, January 2006.

[6] Hello AS. Hello homepage. http://www.hello.no/. Accessed 19th July 2006.

[7] G. Bianchi, F. Formisano, and Giustiniano D. 802.11b/g link level measurements for an outdoor wireless campus network. In *IEEE Proceedings of the 2006 International Symposium on a world of Wireless, Mobile and Multimedia*, 2006.

[8] Atheros Communications. Atheros customer products database
. http://customerproducts.atheros.com/customerproducts/. Accessed 30th August 2006.

[9] T. Friedman, A. Clark, and R. Caceres. RTP control protocol extended reports (RTCP XR). *RFC 3611*, November 2003.

[10] G. Q. Maguire Jr. Personal communication with professor g. q. maguire jr., October 2003.

[11] Mathew Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly, New York, 2002.

[12] Olof Hagsand. Wireless tools for linux. http://www.nada.kth.se/˜olofh/sphone. Accessed 2nd September 2006.

[13] A. Hasswa, N. Nasser, and H. Hossanein. Generic vertical handoff decision function for heterogeneous wireless. In *Proceedings of the Second IFIP International Conference on Wireless and Optical Communications Networks WOCN 2005*, pages 239– 243, March 2005.

[14] International Telecommunication Union. Methods for subjective determination of transmission quality. Recommendation P.800, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, 1996.

[15] International Telecommunication Union. Pulse code modulation (pcm) of voice frequencies. Recommendation G.711, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, November 1996.

[16] International Telecommunication Union. One-way transmission time. Recommendation G.114, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, 2000.

[17] International Telecommunication Union. Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrow band telephone networks and speech codecs. Recommendation P.862, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, 2001.

[18] Eberspächer J. and H. Vögel. *GSM Switching, Services and Protocols, second edition*. Wiley and Sons, New York, 1999.

[19] J.C. Converting Signal Strength Percentage to dBm Values. *WildPackets Inc.*, November 2002.

[20] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. IEEE 802.11 rate adaptation: a practical approach. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 126–134, New York, NY, USA, 2004. ACM Press.

[21] Ian Marsh, Björn Grönvall, and Florian Hammer. The design and implementation of a quality-based handover trigger. In *Proceedings Of The 5th IFIP-TC6 Networking Conference*, pages 580–591, Coimbra, Portugal, May 2006.

[22] D. Minoli and E. Minoli. *Delivering Voice over IP Networks*. Wiley, New York, March 2002.

[23] Johan Montelius. GSM Network and Services Lecture notes, 2006.

[24] Cicero Networks. Cicero networks homepage. http://www.ciceronetworks.com/. Accessed 26th July 2006.

[25] Nortel Networks. A Comparison Between GERAN Packet-Switched Call Setup Using SIP and GSM Circuit-Switched Call Setup Using RIL3-CC, RIL3-MM, RIL3-RR, and DTAP. In *3GPP TSG GERAN No. 2*, November 2000.

[26] Victor Nunes. VoIP quality aspects in 802.11b networks. Master's thesis, KTH Royal Institute of Technology. Stockholm, Sweden, IMIT, September 2004.

[27] Jonathan Rosenberg and Henning Schulzrinne. SIP: Session Initiation Protocol. *RFC 3261*, July 2003.

[28] Henning Schulzrinne, Steve Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. *RFC 3550*, July 2003.

[29] J. C Severiano. IEEE 802.11b MAC layer's influence on VoIP quality parameters. Master's thesis, KTH Royal Institute of Technology. Stockholm, Sweden, September 2004.

[30] J. Tourrilhes. Wireless tools for linux. http://www.hpl.hp.co.uk/. Accessed 12th July 2006.

[31] H. Tze. Handoff between VoWLAN and Cellular. Master's thesis, University of Toronto, Canada, November 2004.

[32] Unlicensed Mobile Access (UMA). UMA overview. http://www.umatechnology.org/overview. Accessed 18th July 2006.

[33] J. O. Vatn. An experimental study of IEEE 802.11b handover performance and its effect on voice traffic, July 2003. Technical Report. Telecommunications Systems lab KTH, Royal Institute of Technology. Kista, Sweden.

COS/CCS 2007-08