# Efficient UMTS

Lodewijk T. Smit and Gerard J.M. Smit
CADTES, email:smitl@cs.utwente.nl

May 9, 2003

This article gives a helicopter view of some of the techniques used in UMTS on the physical and link layer. Additional, we indicate how we can adapt a UMTS mobile receiver to the current environment at runtime. This ensures an adequate quality of service with minimal energy consumption.

## 1 Introduction

Rapidly, the world is getting more and more connected. Internet plays a prominent role in this trend. At the same time, the world is getting less and less wired. People are becoming more and more mobile, carrying mobile devices such as laptops and mobile phones around. The combination of both trends results in a huge demand for wireless communication.

When we consider current mobile devices as laptops, mobile phones and all other kind of gadgets people are carrying around some painful shortcomings becomes clear.

First, a mobile device is by definition too big and too heavy. You do not want to lug with all that heavy equipment. The source of the problem is mainly the battery. About 60% of the size and the weight of a mobile device is due to the battery.

Indeed, that brings us to the second problem: the battery is too fast exhausted. After a few hours of working, your laptop craves for the main outlet. You can use a bigger battery, but that does not contribute to our first problem. The good news is that the capacity of the battery is increasing every year (more energy for same weight/size). The bad news is that this increase goes obnoxiously slow (about 10% per year). The increase in energy needed for computing power (for additional features) is much faster then the increase in battery capacity, things become worse in the future.

The third problem is the dynamic environment of a mobile device that may change dramatically in the short term as well as in the long term. You are calling with your mobile phone, you walk around the corner and... the connection is broken...

The fourth problem is more an indirect problem for the consumer. More and more standards for wireless communication are coming up, new standards become complexer, have more options and require more tuning. Therefore, the lifetime of devices become shorter and the time to market has to be faster. These conflicting issues tend to make a manufacturer crazy. It has to produce faster complexer devices and services with more features. And last but not least, it should also be cheap! Ultimately, at the end of the chain, the consumer will pay for these disasters in one way or other.

It is clear that there is room for improvement. To summarize, we want an energy efficient, flexible architecture that provides an adequate quality of service. Unfortunately, increasing the capacity of batteries is not the area of expertise of computer science. That is more expertise in the area of, let us say, chemistry. Therefore, we do not search for 'more' energy, but we search for ways to use the existing energy budget in a more efficient way. Or stated better, in a more effective way. Do more with the same amount of energy. But how to achieve this? Well, you are on the right spot. Within our computer science building, on the fourth floor, people are tackling this problem within the Chameleon project. This project is funded

by STW Progress. The approach of the Chameleon project is to use reconfiguration at different levels of the architecture to adapt — like a Chameleon — the mobile device perfectly to the current situation. So, given the QoS requirement of the user and the current environment we adapt *everything* — from hardware to application software — to fit exactly to the user's needs.

Most people can imagine what low-power hardware means. But energy-efficient software? How can you achieve energy efficiency at higher levels? We will give a real world example. We will show how to make a UMTS receiver more energy efficient. But first, we will have an intermezzo and explain in the next section a little bit about the technical background of UMTS so that you are able to follow the rest of the story.

# 2 UMTS Background

This section explains some of the basic principles of the physical and link layer of UMTS. The full specification of UMTS can be downloaded from www.3gpp.org. However, the specification covers thousands of pages and the database that describes the status of the separate documents is already 47 Mbyte, so be warned! Because we are mainly interested in the mobile, we consider only the down-link connection (connection from base station to mobile receiver). Especially for data, the link is asymmetrical. The UMTS terminal will receive more data from the base station than it will transmit to the base station. So, we focus on down-link reception.

## 2.1 WCDMA

The ultimate goal is to send a bit from the base station to the mobile receiver and — even more important — to receive this bit correctly. Beside all other kind of distortions of a wireless channel, one of the problems is that you have to share the wireless link with other people. A well known technique (used in GSM) is to use Time Division Multiple Access (TDMA). So, users transmit in sequence and only one user is sending at the same time. Therefore, the users do not disturb each other. Another technique is that users transmit on a different frequency, so called Frequency Division Multiple Access (FDMA). Using this technique, users may send at the same time, but do not 'see' each other. UMTS utilizes another technique: Wideband Code Division Multiple Access (WCDMA). Using WCDMA, users send at the same time at the same frequency but all using a different associated code. In analogy, all transmitters (musicians) transmit the sound of an orchestra and the receiver's objective is to recognize one of the instruments and to filter out this specific instrument. The technique to do this is called spreading. The process of transmission of a bit is as follows (see Figure 1 for an example):

1. The bit is modulated. A '1' and '0' are modulated to a '1' and '-1' respectively.

2. The modulated bit is multiplied by a so called spreading code that is a vector of so called chips $\in \{-1,1\}$. The length of the spreading code is called the spreading factor (SF). The multiplications of the bit with the chips of the spreading codes are very simple. If the modulated bit is '1', the result is equal to the spreading code. If the modulated bit is '-1', the result is equal to the negated spreading code. The result of the multiplication is a sequence of chips. The number of chips to transmit one bit is equal to the spreading factor.

3. The chips are transmitted over the wireless channel. Additional techniques such as pulse shape filtering and modulation are required, but they are not explained here because they are irrelevant for understanding the basic principles.

4. The received chips are correlated with the same spreading code as used by the transmission. Correlation means multiplication of the spreading code with the received chips and adding all the products. When SF products are added, the result is a soft value that is used and the process starts again with the sum initialized to zero. A high (or low) soft output value means that a '1' (or a '-1') was transmitted with a high probability.

Note that the received chips are not exactly $\in \{1,-1\}$ but can be higher or lower due to the disturbance of the wireless channel. The disturbance of the channel is a result of many different effects, among others: noise, interference between symbols, interference between different reflections, not perfect orthogonal spreading codes due to e.g. differences in arrival time of different signals etc.

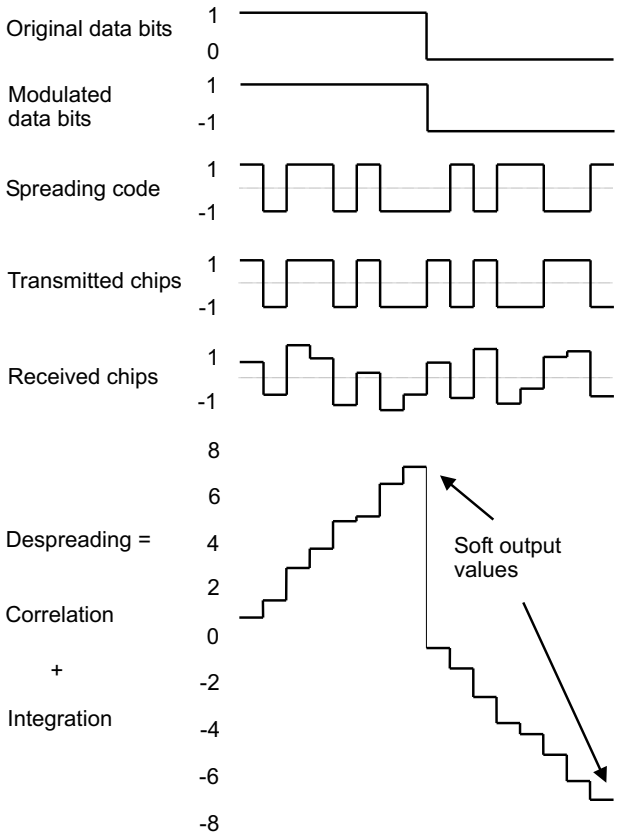


Figure 1: Spreading principle of WCDMA

The secret is that the spreading codes for different users are orthogonal. For example, two orthogonal codes are:

| | | | | | |
|---|---|---|---|---|---|
| spreading code user 1 | 1 | 1 | -1 | -1 | |
| spreading code user 2 | 1 | -1 | 1 | -1 | $\times$ |
| result of multiplication | 1 | -1 | -1 | 1 | |

When the two spreading codes are multiplied with each other and the resulting chips are added, the result will be zero. This means that different users do not disturb each other. UMTS can use different spreading factors, between 4 and 512. This means that for 1 bit between 4 and 512 chips are transmitted. The higher the spreading code, the better the resistance against other users and other external influences that makes the quality of the wireless channel worse. UMTS always operates at 3.84 Mchip/s. Increasing the spreading factor means decreasing the throughput (and the other way around).

## 2.2 RAKE Receiver

The previous section explained that the receiver has to multiply the received signal with the spreading code of the specific user and to add the results of this multiplications. This result is the probability that a '1' or a '0' has been received. However, the receiver used in UMTS operates even smarter. When a base station transmits to a mobile, most frequently different reflections of the signal are received (see Figure 2). In most wireless systems, only the strongest signal is used and the other reflections are considered as distortion and filtered out. UMTS uses a RAKE receiver that rakes (hence the name!) the different reflections together and combines them to make a stronger signal. Figure 3 depicts such a RAKE receiver. A RAKE receiver has multiple so called fingers. A finger can be used to de-spread one reflection. A reflection that travels a longer path has a longer delay. Therefore, the RAKE finger delays the different received reflections until they are synchronized. Using more reflections may improve the signal when the reflections contain enough power. However, it requires more computation.

## 2.3 Forward Error Decoding

Unfortunately, wireless links are not as reliable as fixed links. It is no exception to have a Bit Error Rate (BER) of $10^{-3}$ or even $10^{-2}$. This means that every 1 out of 1000 or 100 bits is received incorrectly. Using a packet size of 512 bytes, this implies that almost no packet will be received correctly. To cope with this,

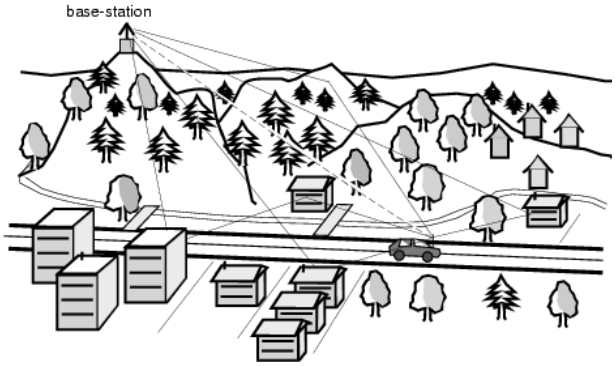


Figure 2: Multiple reflections

UMTS supports two types of forward error correction (FEC) codes: convolutional coding and turbo coding. Forward error correction coding adds redundant bits at the transmitter side, which makes it possible to decode the frame without errors at the receiver side. Using these codes frames can still be received correctly when the channel has a BER of $10^{-1}$. Turbo coding is currently one of the most sophisticated forward error correction codes available. It performs better than (traditional) convolutional codes but requires also more computation power. The FEC decoder is located after the RAKE receiver.

# 3 Adaptivity

In the previous section we saw already a number of choices for parameters. What spreading factor should we use, how many finger should the RAKE receiver use and what kind of FEC coding should we use. In reality, much more options are available, which complicates the exploration of all possible combinations to search for the best set of parameters. At this time, we can start asking interesting questions. When we are not satisfied with the current quality of the output of the RAKE receiver, should we increase the spreading factor or should we increase the number of fingers? Or, maybe, it is better to replace the currently used convolutional decoder (called Viterbi decoder) by a turbo decoder. What is the smartest thing to do? Let us discuss these questions a little bit more structured. First we identify the goal, next we discuss the approach, then we look at the behaviour of the FEC decoders and the RAKE receiver and finally we can draw conclusions.

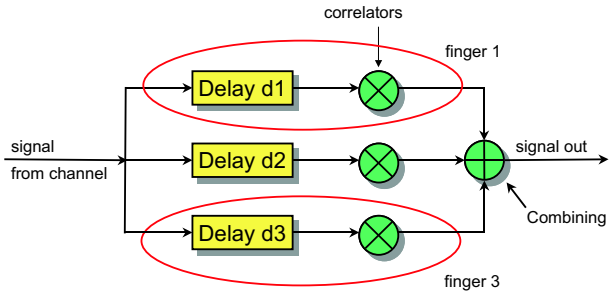


Figure 3: Basic Principle Of A RAKE Receiver

## 3.1 The Goal

Our goal is to select the optimal set of parameters to adapt the receiver to the current environment minimizing the amount of energy at run-time, while still providing an adequate level of QoS. So, our starting point is the requested QoS of the end user. Doing nothing is the most energy efficient way of operation, but does not make an end user happy. Given this QoS constraint and given the current environment we want to minimize the amount of effort to achieve this. Compare it with doing exams, you (probably) minimize the amount of effort to pass it, but you do not want to fail too many times because the amount of effort increases again with re-exams and it makes someone unhappy. So, we try to optimize the parameters to minimize the amount of effort. Thus we perform a functional optimization independent of the implementation and underlying architecture. This should be done at run-time, because the dynamic environment changes continuously. So, a one-time optimization at design time is not possible. Furthermore, the optimization process should be fast and cheap. Of course, the cost of finding the best set of parameters should not exceed the potential saving that could be achieved. And this optimization may not take too long, because the environment could have been changed already again.

Summarizing, we want a control system that se-

lects the best set of parameters at run-time to minimize the energy while still providing an adequate QoS. Figure 4 depicts such a configuration.

## 3.2 Approach

Everything starts with the end user that requested a certain QoS. This may be specified in different dimensions, e.g. the desired throughput, the maximum delay (e.g. are re-transmits allowable?) and the reliability (does the application of the end user tolerate that received packets might still contain errors?) In general, a packet has to be received error free, except for some specific data types used for e.g. video or voice. So, even if the packet contains only one bit error, it is useless and has to be re-transmitted if this is useful anyhow. For streaming data (e.g. video) the re-transmitted packet may arrive too late to be useful. In this case, it is better to save the effort of retransmission. A good quality measure at the application level is the packet error rate (PER).

## 3.3 FEC Decoder Characteristics

A FEC decoder processes usually on a block of bits. How many faulty bits may be received in a block such that the FEC decoder is still able to decode this block correctly? Unfortunately, no nice functions from theory are known that give the error correction capacity of a turbo decoder. To answer this question, we simulated the transmission of blocks under many different circumstances and investigated how often the FEC decoder was able to decode a block correctly as

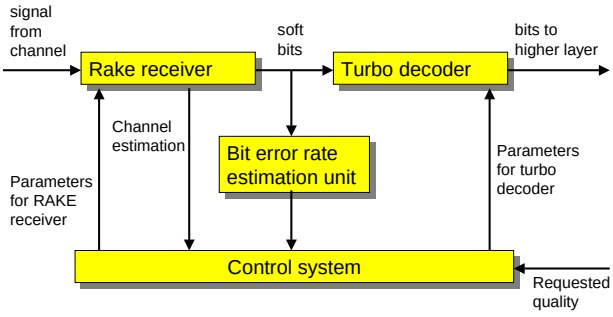


Figure 4: The Control System Of The Terminal

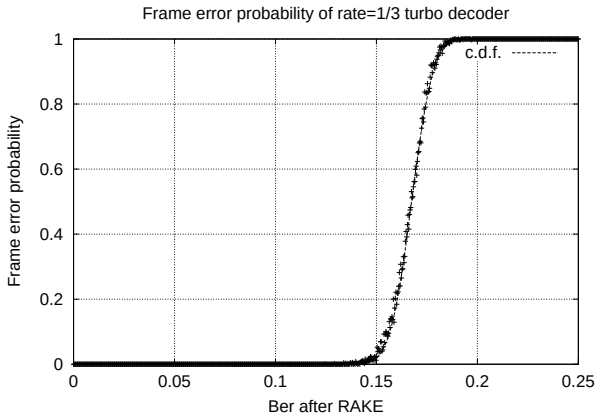


Figure 5: Frame error probability of Turbo decoder with rate=1/3 - 646464 blocks

a function of the BER of the input block. Figure 5 shows the result for a turbo decoder (with some specific settings). So, if the incoming block has a BER of 10%, the turbo decoder is (almost) always able to decode the block correctly. For a BER of 20%, the turbo decoder is almost never able to decode the block correctly. And for a BER of 16.5%, the turbo decoder can decode the block correctly in about 50% of the cases. The same kind of graphs can be obtained for a Viterbi decoder and FECs with different parameter settings.

Irrespective whether the BER of the input block is 1% or 10%, in both cases the turbo decoder is able to decode the block correctly. Therefore, it is useful to tune the RAKE receiver in such a way that the output of the RAKE receiver is just good enough so that the turbo decoder is able to correct these blocks correctly. Spending more effort in the RAKE receiver to produce a better output is a waste of effort.

## 3.4 Performance of RAKE Receiver

Given the results from the previous section, it is interesting to know the BER of the RAKE Receiver for a specific situation. Given this BER, we can predict whether the FEC is able to decode the received block correctly or not. Knowing the BER at the receiver side is not trivial, because - of course - you do not

know what kind of data is transmitted.

UMTS utilizes pilot symbols to discover the BER. Pilot symbols are a sequence of bits that are predefined. So, this sequence is known at the transmitter as well as at the receiver side. With the reception of this well known sequence of pilot symbols, the receiver is able to compute the BER.

We developed a new algorithm to estimate the BER. This algorithm uses the real data and does not need any pilot symbols. This has two advantages. First the overhead of transmitting pilot symbols is avoided. Second, much more data is available and more samples lead to better statistics. The most important advantage of our method is that we are able to predict what will happen with the BER when we plan to change parameters. For example, we can estimate what will happen with the quality when we add an extra finger to the RAKE receiver or what will happen when we double the spreading factor. We will not bother you with the mathematics behind the algorithm.

Figure 6 depicts several parameters settings for a RAKE receiver for a certain situation. The horizontal axis shows the quality (expressed in BER) and the vertical axis shows the costs. So, for every set of parameters, it depicts the relation between quality and costs. Note that when the channel changes, the quality will differ for the same set of parameters, so the crosses move in the horizontal direction. Each cross depicts a certain set of parameters (number of fingers, spreading factor, number of simultaneously transmitting users, etc.). Next to each cross the spreading factor is shown. Crosses which use the same number of fingers are connected by a line. Two different external situations are shown: one for 6 users and one for 24 users.

We know the limit of the turbo decoder, which is displayed with a thick vertical line. We must stay at the left side of the thick vertical line, otherwise the turbo decoder is not able to decode the block correctly. So, the task is to select the set of parameters with minimum cost with a quality that is to the left of the thick vertical line. With the use of our BER estimation algorithm, we can predict what will happen with the quality when we hop between different set of parameters. The costs for the algorithms are also known and independent of the quality of the channel. Because we know the behaviour of the whole system, selection of the right set of parameters is now piece of cake.

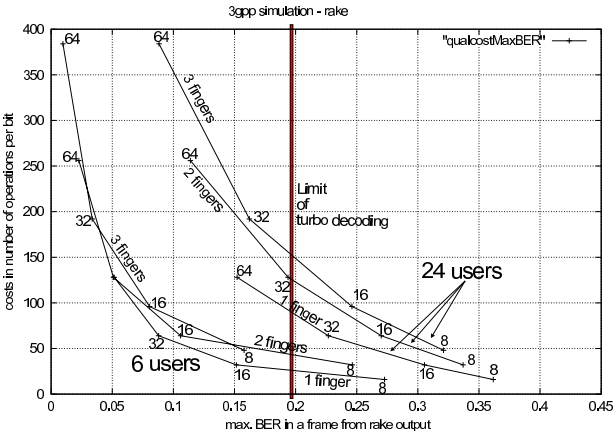


Figure 6: Different parameter settings

# 4 Concluding Remarks

Although the example of UMTS might appear to be very specific, we also showed that the same methodology with only minor adaptation can be used for other standards, like HiperLAN/2 and the new 54Mbit/s IEEE 802.11g wireless LAN standard. These standards use quite different wireless link technologies for the physical layer, with different modulation schemes and using Orthogonal Frequency Division Multiplexing (OFDM) instead of WCDMA.

Interested people are invited to visit us for further explanation or doing their master assignment. A wide range of subjects - from physical level to application level - are available in the area of energy efficient wireless communications. Also check the Chameleon website: http://chameleon.ctit.utwente.nl. See you!