UMEÅ UNIVERSITY
Department of Computing Science
Master Thesis

# Wireless Wilma client

by

Peter Kemppe

Internal supervisor: Thomas Johansson

External supervisor: Thomas Wikström

## Abstract

Wilma is an alarm system where they at monitoring centres receive alarm calls from care phones located in customers home or in shelter housing. As the system works today the operators receive these alarm calls at their work stations, where they are able to take care and respond to these alarm calls.

This report investigates the possibilities of using a handheld alarm receiving client at the monitoring centre. This includes looking at possible protocols and what hardware that could be used as the handheld alarm receiving client.

The security aspects by using a wireless client will also be described.

The report ends by documenting the work of the implementation of a prototype of a wireless alarm receiving client.

# Contents

# 1 Introduction

The use of wireless communication, such as GSM, GPRS and WLAN has been growing very rapidly during the last years. The number of products is increasing at the same time as the price of the products is decreasing. As a result of this the numbers of wireless services are of course also increasing. This thesis investigates some protocols, techniques and products that could be used to extend an existing alarm communication system with a wireless part.

## 1.1 Background

Technology Nexus AB in Umeå (Sweden) develops and maintains among other things care phones and alarm communication systems. The care phones are used primarily by the elderly or disabled who live independently or in a shelter housing. A monitoring centre receives the alarm calls from these care phones and the operator can communicate with the customer through a client/server application that Nexus has developed. The care phone is located in the customers' home or at a nursing home, and it has a unique number that is used as an identifier of the customer. This identifier makes it possible to store information about the owner of the care phone in a database, such as sickness, information about relatives and so on. The operators at the monitoring centre then have access to that information which makes it easier for them to call for proper help (medical help, relatives, etc). One big disadvantage of the system of today is that the operators have to be present at their workstations at all time to be able to take care of incoming alarm calls. Today the operators only have pagers that can notify if there has been a new incoming alarm call, when they are away from their workstations. The information these pagers give is very limited and it only notifies the operator about the alarm, but does not offer any functionality to take care and respond to the alarm call, i.e. they do not have access to the information about the customer and they can't talk with the customer if they so wish.

The fact that they are so tied to their workstations can be a problem at some monitoring centres which do not have very large crews, especially at certain hours of the day. One solution to this problem would be a handheld alarm receiver which communicates wirelessly, and offers functionalities that makes it possible for the operator to respond to the alarm calls in the same way as he/she would do at a workstation. Then the operator would not be as tied to the workstation as he/she is today.

## 1.2 Task description

The goal of this thesis is to investigate different software, hardware, protocols and techniques that could be used to solve the problem described above, i.e. the operators shold be able to take care of incoming alarm calls without having to be located at their workstations all the time. We need some kind of handheld client that can communicate wirelessly with the server at the monitoring centre using GSM/GPRS, WLAN or another existing protocol.

When the above is done a prototype of a wireless client is going to be implemented. It may also be necessary to make some modifications of the server, to enable communication with this new type of client.

# 2 System description

This alarm system is in use in most part of Europe and also in Japan. In Europe the system is used in Sweden, Denmark, Finland, Norway, Iceland, Germany, Austria, Switzerland, Spain and Great Britain. This section will briefly explain how the system works and what the infrastructure looks like.

The care phones that are located at patients' homes communicate with the monitoring centre via the public telephone network. Care takers at shelter housing use these care phones to communicate directly with someone in the staff at the shelter housing via an internal system. But these phones can be configured to automatically call a monitoring centre if there is no answer, within the internal system, after a certain time intervall.

The goal with this thesis is to come up with a solution with a wireless alarm-receiving client for the operators at the monitoring centre, but it may also be interesting in the future to use a similar solution in the internal system. Since in the internal system the staff only communicates with a regular phone and they have no access to any data information about the 'patient' that made the alarm call.

## 2.1 The Wilma application

Wilma is the name of the server/client application that is used within this alarm system. The server application receives the alarm calls from the LU (Line Unit), which it communicates with. The server works as a master unit and the clients works as slave units. The communication between master and slave units is implemented using *Named Pipes* [MSD03], which is discussed more in section 7.2.

The Wilma application is implemented to work as both server and client, i.e. it is eihter configured to work as server or it is configured to work as client. If the application is configured to work as server it will also start a client that will connect to the server part. This makes it possible to take care of incoming alarm calls at the master unit. If the application is configured to work as client it will only run as client, then it will connect to a certain master. The server it tries to connect to also has to be configured.

It is the master unit that runs the database with all customer information. Everytime some kind of information is requested by the client it sends a message to the server which then responds with the data that the client requested.

## 2.2 Monitoring centre

At the monitoring centre there can be at most four operators that take care of the incoming alarm calls, but most of the monitoring centres today have less than four operators. It may also be different number of operators at different times. The reason why there can be at most four operators is that the LU only has four voice communication channels, so there can actually be more than four operators but only four of them will be able to communicate with the customers.

The monitoring centre that has most customers today is located in Copenhagen and has almost 20000 customers. But most of the monitoring centres don't have that many customers. There are those that just have a couple of hundred customers, then it's often just one or two operators at the time that are working. When there are that few operators it is critical to leave the workstation, an urgent alarm call can be made during the time an operator is away from the workstation.

When the alarm button on a care phone is pushed the care phone automatically calls the monitoring centre. The alarm call goes directly to the LU at the monitoring centre through the public telephone network, the LU passes the call through to the server, e.g. the master unit. Then the server broadcasts the alarm to the workstations that are running Wilma as a client, e.g. slave unit, used by the operators. When one of the operators has accepted the alarm the server notifies the other clients that it has been taken care of. The server also tells the LU to open a voice channel between the operators phone and the care phone that made that alarm call. Figure 1 gives a simplified picture of what the infrastructure of the alarm system looks like.

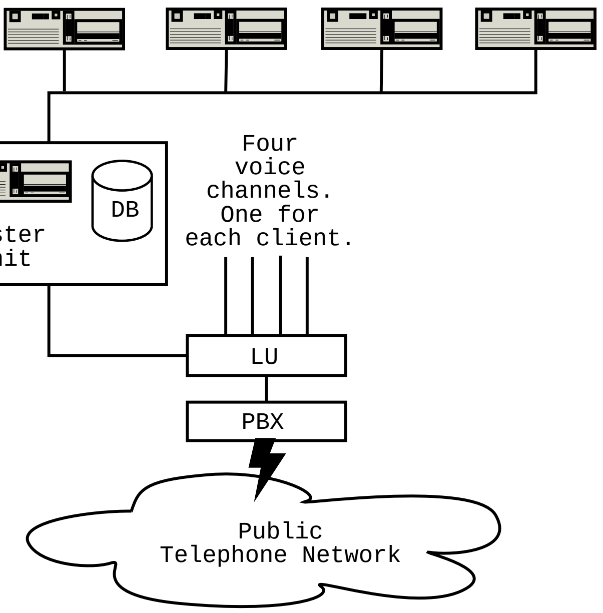

Figure 1: Infrastructure of the alarm system.

### 2.2.1 Types of alarms

There are several different types of alarms. There are regular alarms, battery alarms, lost client alarms and many more. Some of them are more critical than others, e.g. a battery alarm doesn't have as high priority as other kind of alarms. It is especially the alarm calls with high priority that the operators should be able to take care of even when they are not located at their workstation.

### 2.2.2 The care phones

There is a range of different alarms sending devices available, but the most common ones are the Caresse and Camillo which both are care phones. The Caresse is used by the elderly or disabled that live independently and the Camillo is the one that is used by patients in shelter housing.

With the Caresse care phone, all you have to do is push a button to make a call to a monitoring centre, relative or spouse. The care phone is preprogrammed to call certain numbers in a predetermined order. The Caresse care phone also works as a regular phone for incoming calls. Incoming calls is answered with a single push of a button. There are also a range of portable alarm buttons available which can be placed around the wrist or neck, so it is possible to make an alarm call even when you are not next to the care phone. The alarm button communicates over radio with the care phone.

The care phone contains several safety functions. The two most important are the automatical battery surveillance, which sends a battery alarm when the battery level is too low and a test function which sends a test alarm at least once a day to make sure that the communication is working[1].

[1]If the server at the monitoring centre doesn't receive a test alarm from a care phone within a predetermined time interval, then the operators will get notified of this and can make sure that possible failures with the care phone will be fixed.

# 3 Requirements

This section describes the most important requirements that have to be considered during the work with this thesis. Especially when the choice of device and protocol is going to be made.

## 3.1 The wireless device

The existing Wilma client offers the functionality to change customer information and other data in the database. This functionality will not be necessary to have on the handheld device and is just one example of a funcionality that is not necessary. All administration of the database will be done on the workstations just as is it works today.

The wireless alarm-receiving client should be a device that is easy to carry around. The functionality that the device should offer is that it should be able to receive and take care of incoming alarm calls. Of course it will not be possible to implement all the functionality that the existing Wilma client offers. This is, among other things, because of the limitation of memory and display size on handheld devices. The device should have a fairly big display so that we can present as much information as possible about the alarm and the customer that made the alarm call. The possibility to have a voice communication with the customer through the wireless device is also desirable.

## 3.2 The range of the air protocol

The goal with the thesis is to come up with a solution so that the operator doesn't have to be located at the workstation all the time. It should be possible for the operator to leave the workstation for some reason but still be able to take care of incoming alarm calls. The main goal is that the operator should be able to at least move around in the monitoring center building and receive incoming alarms on the handheld device. It is not necesarry to come up with a solution which makes it possible to be far away from the monitoring centre building. I.e. if there is a single operator working at the monitoring centre, then he/she should be able to leave the workstation to take a break and still be able to take care of urgent alarm calls. Another scenario is when there is only one operator at a workstation and one or more are at break. Then if the load of incoming alarm calls is getting high then the operators that have their break can take care of some of the alarm calls to ease the load of alarm calls that are in queue. There are several scenarios where it would be very useful to have such a wireless alarm receiving client, so all of them will not be mentioned here.

## 3.3 Security requirements

The fact that there is sensitive customer information that is being transmitted from the server to the clients makes it very important to think about all the possible security threats. Wireless communication is facing more security threats than wired communication does. So depending on what protocol that is going to be used, different ways of securing the system will be investigated. The security is discussed in section 6.

# 4 Wireless alarm-receiving client

There are several products available today that could work as alarm-receiving clients. Products that may be interesting to take a closer look at are mobile phones that support Java-applications and PDAs (Personal Digital Assistants) running Windows CE, Palm OS or Symbian [Sym03].

There are certain things that we should have in mind when we are looking for products that may work as a wireless alarm-receiving client. E.g. all the data and information that the operator receives must be presented in such a way so that it is easy for the operator to read the information and perform the actions that may be necessary for that specific alarm. I.e. it should have a display that is large enough and has an acceptable resolution so that there will be no problem reading the information that will be presented on the display.

It should also be small and portable. A laptop is portable, has a big screen and would offer all the functionality that an ordinary PC would offer. We only need to install the Wilma application and a WLAN (Wireless Local Area Network) card. But that is not the solution we are looking for. A laptop is not small enough and not as portable as a mobile phone or a handheld computer, which would be more suitable in this connection.

This section describes the possibility of using a mobile phone or a handheld computer such as a PDA as an alarm receiving client.

## 4.1 Mobile phones

Most of the mobile phones that are available today do not have big displays, so there is a limitation of how much information that is possible to present to the operator if a mobile phone is used. There are of course phones available that have quite big displays compared to other phones. One example is the *P800* from *Sony Ericsson* which has a screen resolution of 208x144, (40 x 28 mm), with the flip closed and 208x320, (40 x 61 mm), with the flip open. The *P800* is not just a phone it is also a handheld computer, such devices are described in section 4.3.

Apart from the display size the storage capacity is important, because we have to be able to store the Wilma client application on the device. All the customer information etc. is not that important, because that is stored in the database at the master unit, the only information that needs storage space is the information that is currently presented on the display. But it would maybe be interesting in the future to be able to cache some of the information on the device to improve the performance.

The following information has been taken from the Swedish web-site of Nokia [Nok03]:

> ***How many applications can I store in my Nokia phone?***
> *Nokia standard phones, such as Nokia 3410, Nokia 6310i and Nokia 7210 has 120-150 KB memory for storing Java-applications. The size*

*of java-application for download is limited to a maximum of 30 kb, which leads to that we can have up to 4-5 Java-applications stored in the phones memory. If the downloaded files are smaller then the user can store more applications. Users of Nokia 7650 can store much more applications because they can be stored in the same memory that is used to store photos and MMS-messages.*

It is very important that we do not, in a early stage, get limited by things like memory. Although it is hard to estimate the minimum storage capacity that will be needed.

If the application should be implemented with Java for so called Java-phones, then it is still possible to choose another device to act as the alarm receiving client as long as the device support MIDP (Mobile Information Device Profile). The Java API and runtime environment for embedded systems is described in section 4.4.

## 4.2 Handheld computers

There is a range of handheld computers (handheld PDAs) today from many manufacturer. PDAs have bigger displays than most of the mobile phones that exist today and most of them have pretty good screen resolution compared to mobile phones. The most common display size is 3,5” with resolution 240x320 and has 65536 colors. It is also possible to find PDAs today that have built in Wi-Fi[2] (802.11b) connectivity. And for PDAs that do not have built-in Wi-Fi there are several vendors that provide Compact Flash and Secure Digital cards that makes it possible to connect it to an IEEE 802.11b network. This gives the opportunity to get a connection at 11 Mbit/s or 22 Mbit/s depending on what access point (AP) that is being used. Most PDAs also have built-in speaker and microphone, which is good when it comes to voice communication. There are also accessories for PDAs such as add-on modules and cards available for GSM/GPRS connectivity. See section 5 for more information about the different protocols.

Handheld computers have greater storage capacity than mobile phones. Most of the PDAs today have 64 MB ram and it is also possible to increase the storage capacity with a memory card.

An example of a PDA that has built-in Wi-Fi connectivity is the *iPac 5450* form *HP*. Apart from Wi-Fi it also supports IR and Bluetooth. It also has biometric security using fingerprint recognition and the storage capacity is 64 MB.

## 4.3 Combined phone and computer

There are also several products that work as both mobile phone and handheld computer. There are PDAs with built-in mobile phone and there are mobile phones

[2]Wi-Fi is a label for WLAN products that is based on the IEEE 802.11b standard. A Wi-Fi labelled (or Wi-Fi certified) product is guaranteed to work with other products that have the Wi-Fi label.

with built-in PDA. As mentioned before, the *P800* from *Sony Ericsson* is a mobile phone that also is a handheld computer. The *P800* runs with Symbian OS 7.0 [Sym03] which support applications based on Java and C++. It is of course good if there is more than one choice when the decision is made of what programming language that is going to be used when the application is going to be implemented. Further the *P800* has a storage capacity of 12 MB which is much more than the Nokia phones that were mentioned in section 4.1, on the other hand the *P800* is a computer so the storage capacity should be greater.

Mobile phones are built primarily for voice communication. Because of that the sound quality will be quite good when it is used for voice communication. Voice communication is something that we want the wireless Wilma client to support.

As mentioned in section 4.2 there are several PDAs that have built-in Wi-Fi connectivity and it is possible to get add-on modules for GSM/GPRS connectivity. Apart from that it is also possible to get a PDA that has built-in GSM/GRPS connectivity, i.e. that can be used as a regular mobile phone. One example of such a PDA is the *Qtek 1010* from *Carrier Devices* which is a combined GSM phone and PDA. The OS (Operating System) this PDA runs with is Windows Pocket PC, and apart from GSM it also supports GPRS. This is just one of several PDAs that supports GSM/GPRS connectivity.

## 4.4 Java for embedded systems

The Java standard for mobile phones and entry-level PDAs is called MIDP [SM03c] that is defined by mobile phone vendors and operators. MIDP is just a part of the complete Java because of the limitation of memory in devices such as mobile phones. MIDP is a profile of CLDC (Connected Limited Device Configuration) [SM03a], that is a part of J2ME (Java 2 Micro Edition) [SM03b], which is Java for embedded systems. J2ME is in turn a part of the complete Java.

The MIDP combined with the CLDC is the Java runtime environment for the mobile phones and the entry-level PDAs of today. What MIDP provides is the core application functionality required by mobile applications including the user interface, network connectivity, local data storage and application lifecycle management - packaged as a standardised Java runtime environment and set of Java APIs. Sun offers two virtual machines that support the CLDC, the K virtual machine (KVM) and the HotSpot virtual machine.

KVM is a virtual machine that is suitable for devices with 16/32-bit CPUs and with 160 KB of total memory available for the Java technology stack. 128 KB of this is for the storage of the KVM and libraries, the remainder is for Java applications.

HotSpot virtual machine is for newer devices with larger available memory. This implementation is suitable for devices with 32-bit CPUs and with 512 KB to 1 MB of total memory available for the Java technology stack including applications.

## 4.5 Summary

PDAs has bigger displays and greater storage capacity than a mobile phone. Since it is possible to get a PDA that has GSM/GPRS connectivity it would be a wise decision to choose a PDA. A PDA running an OS such as Pocket PC 2002, will provide more options when we decide what programming language to use. We had of course been able to choose between Java and C++ if we had choosen a phone running Symbian OS 7.0, such as the *P800* from *Sony Ericsson*. But another choice of phone could have forced us to implement the application using MIDP. Because of the great storage capacity, compared to mobile phones, PDAs even make it possible to use J2SE (Java 2 Standard Edition) when implementing the application. Therefore we made the decision to use a handheld computer as a wireless alarm receiving client.

# 5 Protocols

This section describes some air-protocols. The protocols that this section describes are GSM/GPRS and WLAN (IEEE 802.11). Because Bluetooth have more limited range it will not be described here and it will not be used in this work.

The transfer rate of the different protocols will also be looked at. Maybe the transfer rate is not that important because the amount of data that are being transmitted between the server and clients are not that much. It is of course more appealing to have a high transfer rate than a low one. A high transfer rate can maybe be justified if it is necessary to have voice communication with the customer via the wireless client.

## 5.1 GSM/GPRS

GSM is using FDD (Frequency Division Duplex) for uplink and downlink [Sch00a]. In GSM 900 there are 124 channels, each 200 kHz, in both directions. Each of the 248 channels is then separated in time via TDMA (Time Division Multiple Access) frame, i.e. each 200 kHz carrier is subdivided into frames that are repeated continuously. The duration of a frame is 4.615 ms. A frame is again subdivided into 8 time-slots, where each slot lasts for 577 ms and represents a physical channel. Thus, each channel occupies the 200 kHz carrier for 577 ms every 4.615 ms. So 248 channels and 8 time-slots give us 1984 physical channels when including both uplink and downlink.

A GSM system is using CS (Circuit Switched) communication for regular phone calls. For circuit-switched communication, the network sets up air interface connection by allocating one channel (or one time-slot within a channel) to a STA (Mobile Station) when this host wants to transmit data through the network. The radio channel is then occupied by the STA for the duration of the connection. Even if only a small amount of data is transferred, the user has to pay for the duration of the connection.

GPRS (General Packet Radio Service) uses PS (Packet Switched) communication. For PS communication, the network delivers a packet with data when the need arises. Thus, for the air interface, one radio channel can be shared between several STAs simultaneously. In addition, one STA can use up to eight time-slots simultaneously. When a STA generate a data packet, the network forwards the packet to its address on the first available radio channel. Since data traffic often consists of bursts of data, the radio channels will be used efficiently. Address information is included with each packet to enable the packet to find its address. With GPRS the user does not pay for the duration of the connection. The user only pays for the total amount of data that the users STA have sent and received.

In a cell supporting GPRS, the BSS (Base Station Subsystem) maintains two idle lists (the GSM and GPRS idle lists) over all the available channels.

For CS calls, the GSM system will first search for an available channel from the GSM idle list. If no channel in the GSM idle list is available, then the GPRS idle list will be searched. If all the channels are occupied, then a channel in the PS domain will be stolen, because CS communication has got higher priority than PS communication. Once the CS call has got a channel, that channel will be occupied for the duration of that connection.

One big advantage with PS calls is that they can achieve a much higher transfer rate. This is because the STA can use up to eight time-slots simultaneously under a PS communication.

### 5.1.1 Transfer rate

When a user makes a regular phone call, that user is being allotted one time-slot, and no one else can use that time slot as long as the first user occupies it. With GPRS, several users can share one or more time-slots because one user only makes use of the time-slots under short periods of time. This is when there are data packets transferred from the phone or received by the phone. The data transfer rate of one traffic channel in a GSM system is between 9.6 Kbps and 14.4 Kbps depending on what transfer rate the phone supports and what transfer rate the GSM system supports. So if we have a phone and a GSM system that supports 9.6 Kbps, then we can reach a theoretical rate of 9.6 Kbps * 8 time-slots = 76.8 Kbps. But the mobile phones today do not make use of eight time-slots and the number of time-slots that the phone can make use of is different in uplink and downlink. Most of the GPRS phones of today make use of three time-slots for receiving of data and one time-slot for sending data, which is denoted 3+1. There are phones that make use of two or four time-slots for receiving data. So if we have a 3+1 phone and a channel transfer rate at 9.6 Kbps we get a transfer rate of 28.8 Kbps in downlink, but the actual throughput probably doesn't reach that rate. The actual transfer rate for a GPRS connection depends especially on three things:

- **The system**: The transfer rate in one time-slot differs between operators.
- **The phone**: How many time-slots the phone can make use of.
- **Traffic**: If the load on the GSM system is very high in the area where we want to use GPRS, then the transfer rate can be less than the phone actually can handle. This is because regular voice connections have higher priority than GPRS connections in the GSM system, and a GSM connection can take a time-slot from a GPRS connection.

So if a mobile operator offers a transfer rate at 13.4 Kbps per time-slot and we are using a 3+1 phone, then we can send data at a rate of 13.4 Kbps and receive data at a rate of 40.2 Kbps. But the phone can be degraded to use 2+1 or 1+1 depending on how heavy the load is on the GSM system.

### 5.1.2 Quality of Service

Users of GPRS can specify a QoS-profile. This profile determines the service precedence (high, normal, low), reliability class and delay class of the transmission, and user data throughput [Sch00a].

For this alarm system it is very important that we can specify some kind of QoS-profile, which would give us some performance guaranties. I have been in contact with three Swedish operators and no one of them offers any QoS in their GPRS system and all GPRS users share the network equally. None of these operators I was in contact with could say if there was going to be any QoS in their GPRS system in the future. This would be a big disadvantage if we decide to use this protocol for the communication between the wireless Wilma clients and the master unit. We have to be sure that an alarm is getting through to the operator within a certain time from the moment that the alarm button is pushed. This is also the case when the Wilma client sends a reguest for some information stored in the database at the master unit.

### 5.1.3 Realization

With GPRS a user can always be connected without having to pay for the total time of the connection, the user pays for the total amount of data that has been sent and received. Because of this it is important that a user does not receive any data that he/she has not asked for. Because of this it should not be possible to send data to a STA when that STA has not sent a request for that data. According to the three operators that were contacted it is not possible to send data to a STA in their system if the STA has not sent a request for the data. The client would have to poll the server and check if there is any alarm that has not been taken care of.

We should always know if we are able to communicate with the server at the monitoring centre and are able to receive alarms. This could be solved by letting the server send a response message when the client polls the server, then the client would know if it is able to communicate with the server. This would lead to some OH (overhead) in the communication but it is very important though. Maybe it is not necessary for the server to send a response message every time the client polls, but it should respond quite often. If the client has not received a response message from the server in a certain time interval the client has to notify the operator that there is no contact with the server.

If the client is going to communicate with the server at the monitoring centre using GPRS, the client would have to be implemented to poll the server and check if there is any alarm that has not been taken care of as was mentioned before. This is because it is not possible for the server to notify the wireless client for new alarms in the GPRS system. The client has to send a request. Then a decision has to be made on how often the client shuld poll the server. The time interval between two polls can not be too long, because customers may need urgent help. And a very short time interval between the polls would probably make the clients battery consumption increase.

### 5.1.4 Advantages

With a solution using GPRS it would be possible to receive data from the server without having to be within a certain distance from it. GPRS runs over the existing GSM system which allocate between one and eight time-slots within a TDMA frame. The time-slots are not allocated in a fixed, pre-defined manner but on demand and further the uplink and downlink are allocated separately. The active users can share all time-slots.

### 5.1.5 Disadvantages

The fact that the client have to poll the server to check for new alarms is a big disadvantage. It would be much better if the server notifies the clients automatically when there is a new alarm in the alarm queue. Another disadvantage is the fact that a GSM connection can steal channels from a GPRS connection because CS communication has higher priority than PS communication. Further there is no QoS and we do not know if there is going to be any QoS for GPRS connections in the future.

## 5.2 IEEE 802.11

IEEE 802.11 refers to a family of specifications for WLAN. The IEEE 802.11 standard was developed 1997 and it defines three layers. They are FHSS (Frequency Hopping Spread Spectrum), DSSS (Direct Sequence Spread Spectrum) [Sch00a] and IR (Infrared).

There are several 802.11x specifications, but the most common today is 802.11b, also known as Wi-Fi, that a majority of the products today supports. 802.11b uses the DSSS version of the physical layer and it operates in the license free 2.4 GHz ISM (Industrial, Scientific, Medical) band, which is available in most countries around the world.

Since 802.11b is widely used today and there are a lot of available products this section will describe 802.11b networks. The other specifications in the IEEE 802.11 family will not be described here.

### 5.2.1 Transfer rate and range

IEEE 802.11b is known as high rate DSSS and supports up to 11 Mbps. Some manufacturers offer wireless access points that use the 802.11b standard but transmit data at 22 Mbps. These devices will only exchange data at this high rate with other devices built to transmit at 22 Mbps. They are also backward-compatible with standard 802.11b devices, but will only exchange data with them at 11 Mbps. These figures of the transfer rate are the theoretical transfer rate, the actual rate that is possible reach is much lower. There are many factors that affect the transfer rate. The fact that IEEE 802.11b operates in the license free 2.4 GHz ISM band makes that we will get interference from some cordless phones and microwaves. Apart from that there are several other WLAN technologies such as Bluetooth, HyperLAN, etc, that also operates in the license free 2.4 GHz ISM band. The distance between the STA and AP will also affect the transfer rate, because a weak signal will cause retransmission of more packets. Interference or a weak signal causes the data transfer rate to drop to 5.5 Mbps, then 2 Mbps, then 1Mbps. This ensures that the connection stays reliable.

The range that an IEEE 802.11b network provides to the AP (Access Point) is up to 250 meters outdoors with optimal conditions. The range indoors is up to 50 meters with optimal conditions. The transfer rate and range of a wireless network depends much on the environment surrounding it. A signal that does not reach its destination because of interference has to be retransmitted, so high interference will decrease the throughput in the network. Concrete walls, reinforcement bars and micro ovens are just a few examples of things that can cause interference to a wireless network.

### 5.2.2 Quality of Service & Fairness

IEEE 802.11b WLANs provide *best effort service* similar to wired networks. This means that every data packet that is handed over to the 802.11b interface receives the same treatment in terms of delivery guaranties. Thus, it does not receive any Quality of Service guarantees from the network in terms of available bandwidth.

The maximum transfer rate in most wireless networks are much lower than in wired networks, so in wireless networks it is more important that there is fairness in the protocol. That is, every STA should have the same probability to send a packet on the network. CSMA/CA [Sch00a] is the medium access mechanism used in 802.11b which is derived from CSMA/CD [Sch00a] used in ethernet. The goal of CSMA/CA is to provide each STA with an equal opportunity to access the medium. This is done via contention, giving every STA the same probability to send a packet on the network. This is very important when there are many STAs competing for access to the medium. But in our case in the alarm system there will be at most four STA competing for access to the medium, so there will likely not be any problem to get access to the medium. No STA should suffer of any starvation. More STAs can of course try to communicate with the AP, but the AP should be configured to only grant access to the STAs working as Wilma clients. More about that in the security section.

In our case it is more important that we have a good signal and an environment surrounding the WLAN with low interference.

### 5.2.3 VoIP - Voice over IP

Today the operators at the monitoring centre use a regular phone to communicate with the customers. The phone is connected with the LU that opens a connection between the care phone and the operators phone. It is the operator that decide if the LU should open a voice communication channel or not. This is done with a command from the Wilma client application.

One solution to the voice communication problem regarding the wireless Wilma client is to use a regular cordless phone. Every cordless phone then has to be strongly associated with a specific Wilma client. Another solution, when using 802.11b, would be to implement the wireless Wilma client to support VoIP (Voice over IP). Today the voice communication is analog, while data networking is digital. So if there is going to be voice communication with VoIP there will have to be a way to convert the digital data to analog signals and vice versa. Where and how this should be done is not within the area of this work, but it certainly has to be done if the voice communication should be implemented using VoIP. It is the server that is connected with the LU so it would maybe be wise to have a converter there.

All interference that is caused by cordless phones, microwave ovens and more can cause the 802.11b performance to drop [Gei02b]. This is critical for voice communication that need a reliable connection with very little delay. For this it is possible to buy voice engine software that increase the sound quality for VoIP. GIPS (Global IP sound) have a voice engine, *GIPS VoiceEngine*, that they claim can withstand up to 30% packet loss and maintain voice quality. It solves problems like delay, packet loss, jitter, clock drift and acoustic echo. Supported platforms are Windows 98, 2000, NT, Me, XP and Pocket PC 2002. It also includes among other things an API for voice related issues. It might be worth using this, or some other voice engine, if the voice communication should be implemented with VoIP.

### 5.2.4 Realization

If we should use the IEEE 802.11b protocol then we would not have to make the Wilma client poll the server to see if there are any new alarms, as would be the case if we are going to use GSM/GPRS as protocol. We could implement the communication using sockets over TCP/IP (Transmission Control Protocol/Internet Protocol), then we would also get a reliable communication. IEEE 802.11b would give us a limitation in what range the client can be from the AP, but it would provide a much higher transfer rate than GSM/GPRS.

### 5.2.5 Advantages

There should not be any problem finding hardware such as PDA and AP because there is a range of Wi-Fi labeled products available. The implementation of the communiation can be done over TCP/IP just as if it were a wired connection. The MAC (Medium Access Control) layer handles the communcation between the radio network card and AP. The MAC layer uses an 802.11 physical layer such as 802.11b or 802.11g to perform tasks like carrier sensing, transmission and receiving of 802.11 [Sch00a, Gei02a] frames.

### 5.2.6 Disadvantages

The range is a limitation when using the 802.11b protocol. The fact that it is using the licence free 2.4 GHz ISM band makes it suffer from interference as was mentioned before.

## 5.3 Summary

Since we do not want to have a solution where the Wilma client polls the server to check if there are any new alarms the GPRS protocol should not be used. To have the client poll the server all the time would probably also increase the battery consumption a lot compared to if we would use a solution with IEEE 802.11b, where the Wilma client does not need to poll the server. As mentioned before, IEEE 802.11b would also provide a much higher transfer rate than GPRS.

There is a limitation in how far away from the AP the client can be if we use IEEE 802.11b, but we are not looking for a solution where the operator can be far away from the monitoring centre, we are looking for a solution to make it possible for the operator to leave his work station and still be able to take care of incoming alarm calls. That is, the operator should be able to take care of incoming alarm calls no matter where he/she is within the monitoring centre. Therefore the decision was made to use the IEEE 802.11b protocol for the communication.

# 6 Security in wireless networks

The security threats and vulnerabilities that exist for wired networks also exist for wireless networks. And there are more things to think about when setting up a wireless network. The fact that it is not possible to control the distance of the signal, it may travel outside the building where the AP is located. This makes it possible for someone to gain access to the network by simply sitting in a car outside the building with a laptop equipped with a WLAN-card. More about that in section 6.2.

## 6.1 Security threats

When talking about security threats it is often refered to the possibility of some kind of attack. The attacks can be devided in to two categories, *Passive Attack* and *Active Attack* [Kin01]. Below is a description of the different types of attacks that fall into these two categories.

- **Passive attack**
  This is a type of attack where an unauthorized party gains access to some resource and does not modify it's contents. Passive attack can be either eavesdropping or traffic analysis.
  - **Eavesdropping**
    The attacker monitors the traffic for message content. This can be a person listening to the traffic between two workstations on a LAN or the traffic between a wireless device and a AP.
  - **Traffic analysis**
    The attacker monitors the traffic looking for patterns of communication. This could be to gain information to crack a WEP (Wired Equivalent Privacy) key or of some other purpose. The amount of data that is being transmitted between two communication parties contains a considerable amount of information.
- **Active attack**
  An attack where an unauthorized party makes modifications to a message, file or another data stream. An active attack can be one of the four described below, or it can be a combination of some of them.
  - **Masquerading**
    The attacker impersonates an authorized user and in that way gains certain unauthorized privileges.
  - **Replay**
    The attacker monitors the traffic, i.e a passive attack, and records messages. The attacker then retransmits messages as the legitimate user, which is an active attack.

- **Message modification**
  The attacker altering a legitimate message by changing, deleting, adding to or reordering the contents.
- **Denial of Service**
  The attacker prevents or prohibits the normal use or management of a network service.

As Bruce Schneier [Sch00b] says, the solution to these problems are obvious in theory but harder in practice. By encrypting data packets, no one can read them in transit. And by authenticate data packets, no one can insert packets that pretend to come from somewhere else, and deleted packets will be noticed and reacted to.

Because WLANs use radio waves they are much more vulnerable than wired networks are. One of the threats WLANs are facing is that they are the target for drive-by hacking, or war driving, which is an very easy task that does not require any special skills or effort. There is quite many unprotected WLANs in use today. There are many articles about war driving that shows how easy it is to find unprotected WLANs. Just search for *war driving* with a search engine like google and you will get plenty of hits.

There are widely available tools that can be used to listen to the traffic on 802.11 networks. It is important to always have in mind that your traffic can be tapped by someone when you are using a WLAN. Take for example an airport, you arrive at the airport and you want to use the *HotSpot* to download your mail from your mail server to your laptop, or maybe you want to send a e-mail. Then just about anyone having the right tools and knowledge can be listening to your traffic and read your e-mails. Think about all the businessmen that use their laptops at the airport and that have classified information stored on their computers and accidently are sharing their files with everyone who is using the network.

Then we have all the companies that use WLANs in their office building. It is more like putting a internet jack in the parking lot outside so the public can use their network. Many of these companies are aware of all this and spend a lot of money on security but still they can't be sure that their network is not wide open.

## 6.2 Drive-by hacking

If you not secure your wireless network then an intruder can gain access to your wireless network by simply sit in his car outside with a laptop and a PC-card to get access to the network. This is called drive-by hacking or war driving, and a lot of them are done by IT consultants to show the security threats WLANs are facing. And as Jacqueline Emigh says in an article [Emi02]:

> In an Internet forum on ISP-Wireless, a member called "MB" acknowledges, "Being a student, war driving is something we do when we're not partying; we used to drive around and download all night long into our van."

War driving has become a hobby for many people. And as long as the numbers of WLANs are increasing, the interest of war driving will probably not decrease. If there are many WLANs then you do not have to spend so much time to find one.

This made me very curious, I wanted to know if it really was that easy to find WLANs and if there were so many that did not secure their networks. So equipped with a *PowerBook G4* with an *airport extreme* card I and a friend drove around in the streets in Umeå to detect some WLANs. The application that was used was *MacStumbler 0.75b* which sends out probes to APs that are located in the signal range. When a AP receives this probe it answers with a message containing some information about the AP. If the AP answers or not depends on how it is configured, this will be described in section 6.3. If the AP answers to this probe we get the SSID, the MAC-address, information wether it uses WEP or not, the signal strength and more. We were really surprised that we found so many WLANs as we did. We found 56 WLANs and we drove around for about two hours. Among these WLANs only 15 were using WEP, the remaining 41 were wide open. Most of the networks were private home networks but we found a few company networks that were not using WEP. What really surprised me was that we found out that a well known ISP (Internet Service Provider) had a WLAN that was not using WEP. Well, we can not be sure that it really was their network because you never know exactly where the AP is located. But we where driving by their office building when we got an answer from an AP, and the SSID on that AP was the name of the ISP company. So it probably was their network. Maybe they do not use WEP for a reason, but we don't know if that is the case. And I can't think of any reason why they should not use WEP.

## 6.3 How to secure a WLAN

The data that is being transmitted from the server to the client in the alarm system, such as customer information, is highly classified. It is therefore very important that every STA has to be associated and authenticated with the network and that the data that is being transmitted is encrypted so that if someone would sniff the network it would not be possible to understand anything of the data that is being transmitted.

Every Wi-Fi-certified product supports WEP encryption, but WEP is not activated automatically at the installation, this has to be turned on manually. WEP is not secure enough, but there is work in progress to improve the security.

IEEE 802.11 offers three authentication mechanisms:

- **Open System Authentication**
  Only the APs (Access Point) SSID (Service Set Identifier) is used.

- **Shared Key Authentication**
  A static WEP key is used on both the AP and STA.

- **Only accept selected MAC addresses**
  Configuring the AP to only accept selected MAC addresses.

These authentication mechanisms can be used separately or in combination. This is not safe, because there are widely available hacker's tools to go through these. Open System Authentication depends on an attacker not learning the SSID, but this can be learned using a packet sniffer. With Shared Key Authentication, an attacker using a packet sniffer can reuse information from one valid authentication to authenticate himself, and because the key is static and the same for every user it makes it more easy for an attacker to get the key with help of a packet sniffer and a tool such as *WEPCrack*. Finally, the MAC address of a Wi-Fi card can be changed to that of any other (spoofed).

### 6.3.1 Wi-Fi Protected Access

When the AP and STA are configured to use the static key for authentication that same key is used for WEP encryption. This means that an attacker can decrypt the data that is transmitted without having to crack the encryption key, the attacker just reuses the key that was cracked in the first place for authentication.

The Wi-Fi alliance that give WLAN products their Wi-Fi certificate has come up with a new security solution called WPA (Wi-Fi Protected Access). WPA support will be available from vendors of WLAN equipment in early 2003 according to the Wi-Fi alliance. WPA is designed as a software upgrade to existing Wi-Fi certified products so no additional hardware is needed.

WEP requires users to enter a key to authenticate with the AP. This key is then also used to encrypt the data packets sent over the air. Since every user on the same WLAN uses the same key there is no problem for them to sniff each others traffic. With WPA the user start by entering a key, the same way as with WEP, but this key only initializes the encryption process. The actual key rotates all the time and a new key is used for every packet that is transmitted over the network. This means that the users that use the same AP are no longer able to sniff each others traffic. So WPA replaces the static encryption key with a dynamic key. This improved encryption technology is known as TKIP (Temporal Key Integrity Protocol) and is much more difficult to break than WEP.

In a home network or in small offices, where there are no central authentication server, WPA runs in a special home mode. This mode, which is also called PSK (Pre Shared Key), allows the use of manually entered keys. All the users then have to enter this key to gain access to the AP. This keeps out eavsdroppers and other unauthorised users. If a user has entered the correct key (password) in his device he gains access and the key automatically kicks off the TKIP encryption process.

### 6.3.2 Access Point configuration

When configuring the AP there are some simple things that could be done to improve the security on the WLAN [Liu03].

- **Change the default SSID**
  Every AP have a default SSID that is set by the vendor. Hackers know these defaults and can try them to join the network. The SSID is the name of the network and it can be anything you wish. The network's SSID should be something unique, and it should not refer to the network products that are used. The SSID should also be changed on a regular basis, so any hacker who may have figured out the network name will have to figure it out again and again.

- **Disable the SSID broadcast option**
  If the SSID broadcast option is disabled, then it is not possible to discover the network name by using a tool such as *MacStumbler*, as we used. Most APs are by default set to broadcast the SSID, so anyone can join the network if it does not use WEP. Or the WEP key may be cracked using a tool such as *WEPCrack*. So unless the AP is a public hotspot it is best to disable SSID broadcast.

- **Enable MAC address filtering**
  If the products such as AP and router offers it, enable MAC address filtering. The MAC address is a unique series of numbers and letters that are assigned to every network device. With MAC address filtering you configure the AP to only assign access to devices with specific MAC addresses. This makes it harder for a hacker to gain access to the network using a random MAC address.

- **Enable WEP encryption**
  WEP encryption should be enabled, to prevent anyone from sniffing the traffic. The WEP key can be cracked, but it is sure better than no encryption at all. The WEP key should also be changed on a regular basis, so if any hacker has cracked the key, they will have to do it again and again.

- **Disable DHCP**
  DHCP should be disabled and only static IP addresses should be used. In this way a hacker does not get an IP address and has to know a valid IP address that can be used.

Another possibility to protect the data that are being transmitted between the wireless clients and the server in this alarm system, could be to implement the application to use SSL (Secure Socket Layer). Section 7 describes how the communication was implemented.

### 6.3.3 Virtual Private Network

Using a VPN (Virtual Private Network) will secure the network traffic between the wireless client and the edge of the wired network as in this case would be the server that runs the Wilma server application. VPN was developed to enable client systems to securely connect to servers over the public internet. By using VPN you are establishing a secure connection between the clients and a VPN gateway, which in a 802.11b networks, sits just behind the AP. Then each wireless client on the network communicates with the AP over a dedicated VPN tunnel. By placing the VPN gateway behind the AP we can ensure that all communication that goes over the air is protected [Tec03]. Figure 2 shows what the infrastructure could look like when using VPN. If one would use VPN, it would be in addition to WEP.

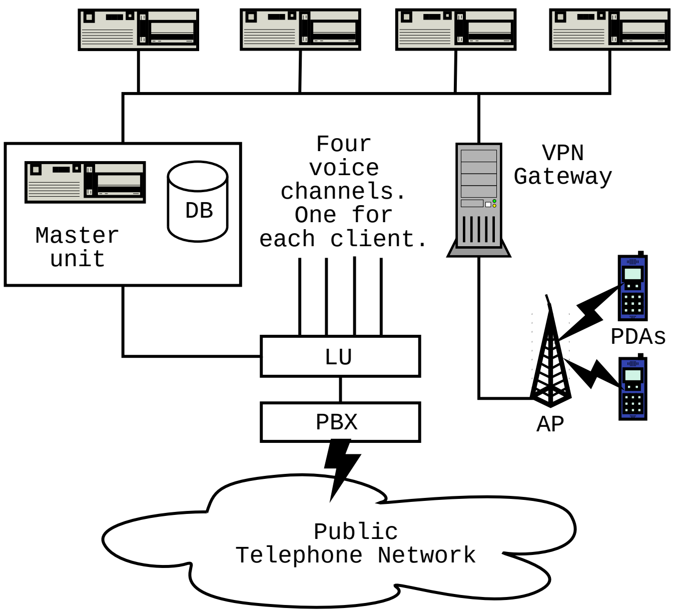

Figure 2: The communication remains strongly encrypted between the clients and VPN gateway, which is located behind the AP.

And it is very important to not get careless. Encryption does not do any good if hackers do find out the key. Likewise, VPN would not succeed if an attacker can gain access to a trusted client [Tec03].

An alternative to VPN is to use an access point with built-in IPsec, available from vendors like Colubris Networks. Their APs solves the WEP encryption problem by using IPsec with 3DES or AES [Kin01].

# 7 Realization description

The hardware that was purchased for this work was one handheld computer and one WLAN router.

The handheld computer is a *HP IPAQ H5450 POCKET PC* that has among other things built-in WLAN (Wi-Fi), 64 MB RAM and 48 MB ROM. Apart from that we got 64 MB of extra RAM memory on a SD-card. The OS on this device is Windows PocketPC 2002.

The router is a *Netgear MR314* which is a router with built-in 4-port switch and integrated wireless IEEE 802.11b AP.

In figure 3 you can see the infrastructure of the system with the wireless part.

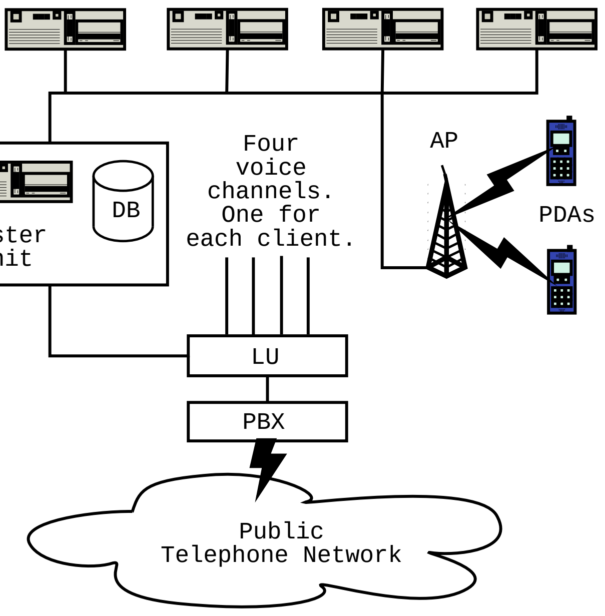

Figure 3: Infrastructure of the alarm system including the wireless part.

## 7.1 Java or C++

The existing Wilma client/server application is implemented with C++, so it would be wise to use C++ instead of Java as programming language when implemeting the wireless Wilma client. This is because it would be possible to reuse classes from the existing client/server application. It would be desirable to reuse as many classes as possible, because it would be stupid to “reinvent the wheel”. The data representation is different between Java and C++, so if we use C++ we will probably not run in to such problems.

We decided that C++ should be used and work with *Embedded Visual C++* when implementing the wireless Wilma client application.

## 7.2 Named Pipes vs. TCP/IP Sockets

In the Wilma client/server application of today the communication between client and server is implemented using *Named Pipes*.

*Named Pipes* and *TCP/IP Sockets* [DC01] are comparable in terms of performance. The difference in performance becomes apparent with slower networks, e.g. across WANs (Wide Area Networks). *TCP/IP Sockets* are preferred in slow LAN, WAN or dial-up networks and *Named Pipes* can be a better choice when network speed is not the issue [Kha01].

The API for Windows Pocket PC does not support *Named Pipes*, so the communication has to be implemented using e.g. TCP/IP Sockets. This will also lead to that the Wilma server application has to be modified to support communication through both *Named Pipes* and *TCP/IP Sockets*.

## 7.3 Wilma server

The Wilma application works as both server and client. I.e. it can either be configured to work as a client or as a server. If the application is configured to run as server it will also run a client that communicate with the server part. If it is configured to run as a client, then it will only work as a client.

As mentioned before the communication between the server and clients is implementeted using *Named Pipes*. And to make it possible for the server to communicate with a device running Windows Pocket PC as OS it had to be modified so it also could communicate using *sockets*. Figure 4 shows a class diagram over the classes that are involved with the communication, all the attributes and methods are not included in the class diagram because there is a lot of them.

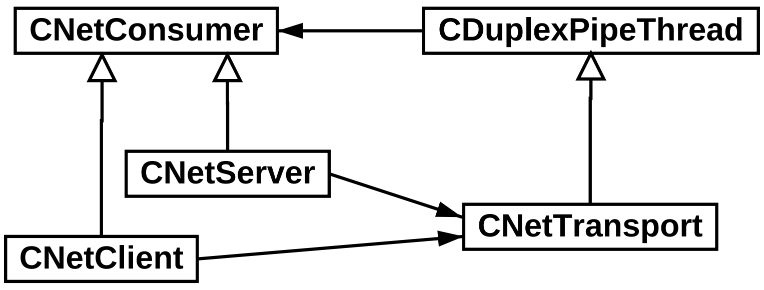

Figure 4: Class diagram with the classes that handle the communication before any modification was made.

Here follows a brief explanation in words how the instances of these classes interact and what modifications that had to be made with the existing Wilma application. The goal was to modify the server so it could communicate with both *Namerd Pipes* and *sockets*. These modifications were made with success and after the modifications it is now even possible to configure the client to use either *Named Pipes* or *socket*. Before a description of the modifications that were made, an explanation of the classes in figure 4 will be given.

- **CNetConsumer**
  This class is pure virtual and it only contains two virtual methods. The purpose with this class is to make CNetClient and CNetServer inherit from it. Then CDuplexPipeThread has one attribute of type CNetConsumer which then either is instantiated to be a CNetServer or a CNetClient.
- **CNetClient**
  Inherits from CNetConsumer. Has one pointer attribute of type CNetTransport which is used for the communication.
- **CNetServer**
  Inherits from CNetConsumer. Has one pointer attribute of type CNetTransport which is used for the communication.
- **CNetTransport**
  Inherits from CDuplexPipeThread and it only contains one constructor that passes all the parameters on to the super class. The purpose of this class is a little obscure, it only contains one constructor which only passes on the call to the super class. It would be possible to replace the CNetTransport attributes in CNetClient and CNetServer with attributes of type CDuplexPipeThread and still have the application working as it does now. Then we could remove this class because it is not in use anywhere else.
- **CDuplexPipeThread**
  Inherits from CWinThread. This is the class that handles all the communication. It has one member varibel of type CNetConsumer which is instantiated to be either a CNetServer or a CNetClient. Has a *HANDLE* to a *Named Pipe* which is used for the communication.

When data is being sent and received the thread method *CDuplexPipeThread::Run()* is using the API functions *WriteFile* and *ReadFile* which among others take a handle to the *Named Pipe* as parameter. According to the API this handle could also be a handle to a *SOCKET*. So the first thought was, this is going to be easy. We only have to create a *SOCKET* instead of a *Named Pipe* and use that instead. So this was done and it almost worked. The case is that when a read operation has been made the application checks if there is more data to be read on the *Named Pipe* or if it got the whole packet. This is done using the API method *GetLastError* which returns the error code *ERROR_IO_PENDING* if not all data were received. The main reason for this check is that the buffer that we are using to store the incoming data can hold 10 kilo bytes of data, which then is the maximum amount of data that will be read in a single read operation. This means, to receive a packet of size 67 kilo bytes at least seven read operations will be needed to get the whole packet. When a read operation has been done and there is more data to be read, the thread method *CDuplexPipeThread::Run()* passes the data to the member method *Insert-PartialRead(BYTE, DWORD)* which temporarily stores the data as a node in a list. When the last read operation is done of a packet the data is being passed on to the member method *InsertReceivedData(BYTE, DWORD)* which then rebuilds the whole packet if there is any data temporarily stored in the list. If the list is empty the packet already is complete.

When the handle to a *Named Pipe* was replaced with a handle to a *SOCKET* there was no problem with the compilation or linking of the application. Unfortunately it did not work as expected. It did not work as it should. The fact was that when a read operation had been made the *GetLastError* method indicated that there was no more data to be read on the *SOCKET* and the whole packet had been received, even if there were more data to be read. This lead to that when another packet arrived at the *SOCKET* the application started to read the remaining of the previous packet and treated this as a new packet. Because of this the application did not work at all. Unfortunately it took quite a while to debug the application and locate where it all went wrong. But finally we printed out the log of the application using *Named Pipes* and the log using *SOCKET* and by comparing these the error was discovered.

Then a decision was made that a new class was going to be implemented that should take care of the communication using *SOCKET*. The *CDuplexPipeThread* class should only be used for the communication using *Named Pipe*.

Figure 5 shows the changes that was made compared to the one shown in figure 4.

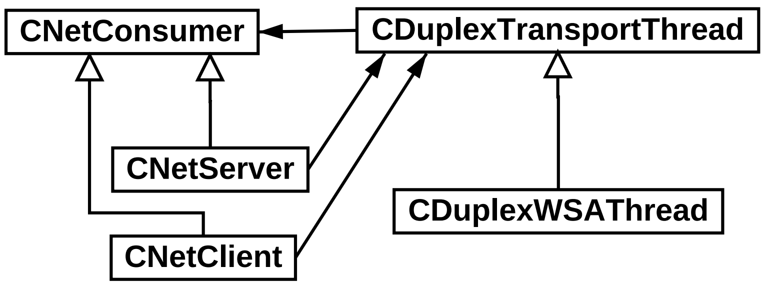

Figure 5: Class diagram with the classes that handle the communication after the modifications were made.

The class *CDuplexTransportThread* is exactly the same as CDuplexPipeThread, it is only the name that has been changed. As was mentioned before, the class CNetTransport was not necessary, so as you can see that class is removed. The class *CDuplexWSAThread* was added which is handeling the communication using *SOCKET*. Why the class name *CDuplexWSAThread* was chosen is because several

WSA-methods are being used, *WSAGetLastError* etc., which are available in the API. *WSASockets* are not used though, *BSD-Sockets* are used. As can be seen in the class diagram in figure 5 the *CDuplexWSAThread* class inherits from *CDuplexTransportThread*. By doing so, only the methods that are going to work differently when using *SOCKET* instead of *Named Pipe* had to be overloaded. Now *CNetServer* and *CNetClient* has pointer attributes of type *CDuplexTransportThread* instead of *CNetTransport*, which was removed. And thanks to the polymorphism in C++ those can be instantiated to point to objects of type *CDuplexWSAThread*, depending on if *Named Pipe* or *SOCKET* should be used. Below is a list of those methods in *CDuplexWSAThread* that were overloaded from *CDuplexTransportThread*, and no extra methods have been added to the *CDuplexWSAThread* class.

- **Open()**
  Sets the *struct sockaddr_in* member variable to the server address. Creates the *SOCKET* and connects to the server. Then the thread method *Run()* is triggered. NB, this method is only called from the client part of the application.

- **Close()**
  This method had to be overloaded because it is a *SOCKET* that is going to be closed not a *Named Pipe*.

- **InitInstance()**
  The communication is based on events, and when using sockets we need a special event that is triggered when data arrives to the *SOCKET*. For this an event was created with the *WSACreateEvent()* method which is available in the API. This event is later being associated with the *SOCKET* and is triggered when there is data to be read on the *SOCKET*. And as being done in *CDuplexTransportThread::InitInstance()*, this method also create events with the *CreateEvent()* method. These events are triggered when either there is data to be sent or if the application is being shut down.

- **Run()**
  This method had to be overloaded because it did not work to use the methods *WriteFile* and *ReadFile* as was mentioned before. When passing a *Named Pipe* as a parameter to the *ReadFile()* method, as in *CDuplexTransportThread::Run()*, it is possible to determine if all data was received or if there is more to be read. This is determined using the *GetLastError()* method which returns the error code *ERROR_IO_PENDING* if there is more data to be read. But here *send()* and *recv()* is used when sending and receiving data. Then it is not possible to know if all data was received or not, by using a method like the *GetLastError()* method. This had to be solved in a different way. Here is a brief explanation of what is done when data are being sent

and reveived [3].

- **Send data**
  When a new packet is going to be transmitted, then first a packet of 4 bytes is being sent that contains the size of the packet that is going to to be sent. This is done so that the receiver will know how much data the next packet contains. Then the whole packet is transmitted using a loop that terminates when the whole packet has been sent.
- **Receive data**
  Here we have a variable *gotPendingRead* of type *BOOL* which initially is set to *false* and indicates if there is more data to be read or if there is a new packet that is going to be received. I.e. when *gotPendingRead* is *false* there is a new packet that is going to be received. When a new packet arrives the first 4 bytes, which tell the size of the packet, is being received. Then *gotPendingRead* is set to *true*, and remains *true* until the whole packet has been received. This is very important because the data has to be passed on to the right member method, i.e. *InsertPartialRead* or *InsertReceivedData* as mentioned before. Figure 6 shows a flow chart of what is happening when the read event is triggered.

### 7.3.1 Start up phase of Wilma

As mentioned before, Wilma is either configured to run as a server or as a client. So depending on the configuration different things will happen when the application starts up. Here follows an explanation of what is happening during the start up phase, and what modifications that had to be done.

- **Wilma server**
  When Wilma starts up to run as server it creates a thread function, *serverPipeThread()*, that is taking care of new clients that are connecting using *Named Pipes*. Now we also need a thread function, *serverSocketThread()*, that is taking care of new clients that are connecting to the server using *SOCKET*. For each new client that is connecting using *Named Pipe*, the thread function *serverPipeThread()* creates a new *Named Pipe*. After that a new *CDuplexTransportThread* object is created by giving the *Named Pipe* as parameter. Then the *CNetServer* object, that is going to take care of the communication with this new client, is created given the *CDuplexTransportThread* object as parameter. When all this is done the *CDuplexTransportThread* object sets its member variabel of type *CNetConsumer* to be the *CNetServer* object that we just created, then its *Run()* method is started and

[3]Since the sending and receiving of data are based on events the *SOCKETs* are created to be non-blocking.

**Variables that are used in the chart:**

```
BOOL gotPendingRead: True if it is a new packet else false.
DWORD remainingData: Number of bytes remaining of the packet.
DWORD BUFSIZE: Number of bytes that we can store in the buffer.
```

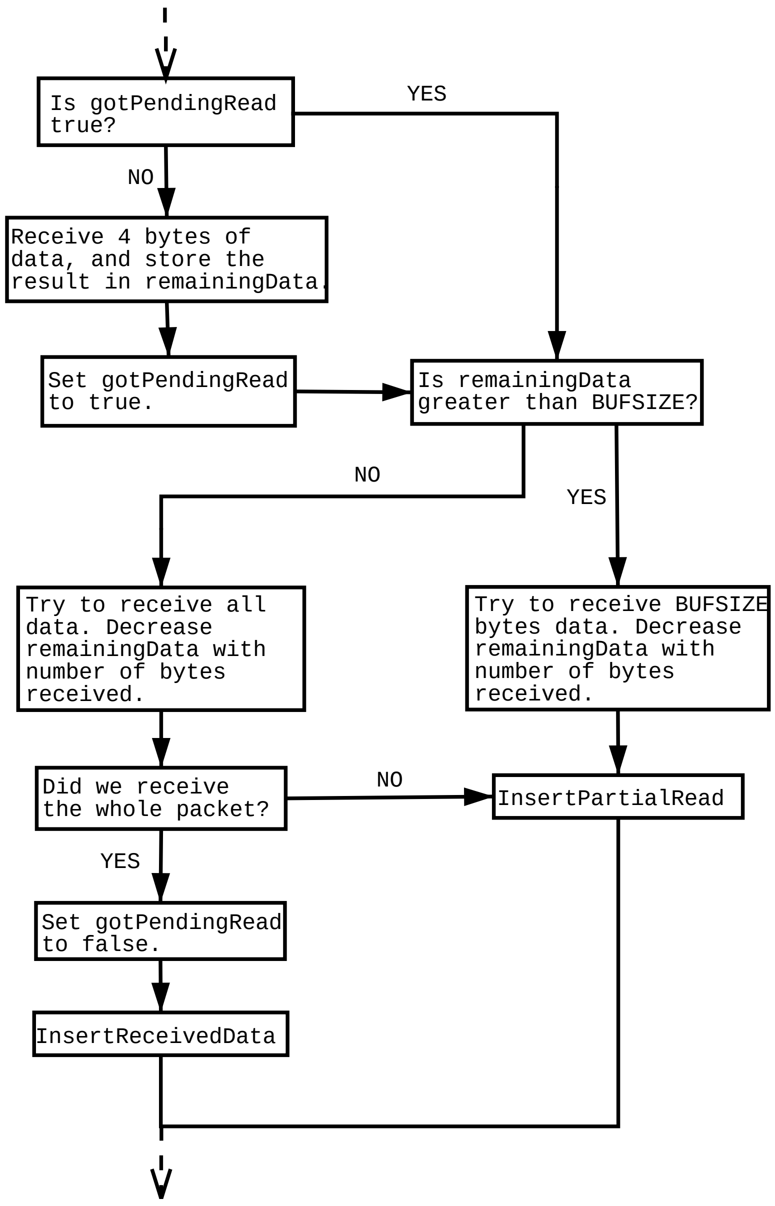
Is gotPendingRead true?
YES
NO
Receive 4 bytes of data, and store the result in remainingData.
Set gotPendingRead to true.
Is remainingData greater than BUFSIZE?
NO
YES
Try to receive all data. Decrease remainingData with number of bytes received.
Try to receive BUFSIZE bytes data. Decrease remainingData with number of bytes received.
Did we receive the whole packet?
NO
InsertPartialRead
YES
Set gotPendingRead to false.
InsertReceivedData

Figure 6: Flow chart of what is happening when receiving data.

the communication can begin. In the new thread function, *serverSocketThread*, we get a new *SOCKET* from the *accept()* function when a new client is connected. And instead of creating a *CDuplexTransportThread* object we here create a *CDuplexWSAThread* object giving the *SOCKET* as parameter. That is the main thing that is different from the thread *serverPipeThread*. To get a better picture of what was just described, a part of the code of *serverPipeThread* and *serverSocketThreag* can be seen in appendix B. The class diagram in figure 5 might also help to get a better understanding of what was just described.

- **Wilma client**
  When the application starts it will start a client part. For this it has a variable of type *CDuplexTransportThread*. And because *CDuplexWSAThread* inherits from that class we can either instantiate this variable to be of either type, depending on if we are going to use *Named Pipe* or *SOCKET*. And then depending of which is being used, either *CDuplexTransportThread::Open()* or *CDuplexWSAThread::Open()* will be executed when it is going to connect to the server. When it is connected the *Run()* method is started and the communication can begin.

## 7.4 Wireless Wilma client

To implement the application for the wireless Wilma client, embedded Visual C++ was used. The goal was to reuse as many classes as possible from the existing Wilma application.

All classes in the wireless Wilma client application, except those that are generated by embedded Visual C++, are taken from the Wilma server application. It was a very time consuming and tricky work to locate exactly what files that should be copied. We only wanted those classes that is used by the client part of the Wilma server application. After a couple of days work it was possible to connect and login to the Wilma server with the wireless client.

Except from copying classes from the server application some modifications had to be done. This is because the API for Windows CE and Pocket PC 2002 is limited. It only includes a subset of the API for applications that is going to be run on a platform such as Windows 2000. This is mainly because of the limitation of memory in devices running Windows CE and Pocket PC 2002.

Figure 7 shows the relation between the classes that handle the communicaion on the client application. If it is compared to the class diagram shown in figure 5, all the changes that have been made can be seen. Here the *CDuplexWSATread* class inherits from the API class *CWinThread*. And because we do not have to use the *CDuplexTransportThread* class, the *CNetClient* has a member variable of type *CDuplexWSAThread* instead of *CDuplexTransportThread*. Also all the methods from the *CDuplexTransportThread* class had to be copied to the *CDuplexWSAThread* class, except those that already were overloaded.

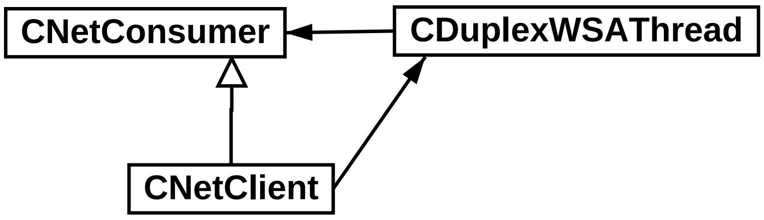

Figure 7: Class diagram of the classes that handle the communication on the wireless client.

The *CNetClient* object is created in the same way as in the server application and it is connecting to the server in the same way as well. One modification that has been done is in the *CDuplexWSAThread::Run()* which handles the communication, which is based on events. In the Wilma server application the API function *WSAEventSelect* is used to associate the *WSAEvent* with the network event *FD_READ* on the *SOCKET*. But because the *WSAEventSelect* function is not avaiable in the API for Pocket PC 2002, it was not possible to use an WSAEvent to let the application react when data arrives at the *SOCKET*.

We still wanted the communication to be based on events, i.e. we wanted to use non-blocking sockets and use the same algorithm as in the Wilma server application. To solve this, all the events were created with *CreateEvent()* in *CDuplexWSAThread::Initinstace()*, i.e. no *WSAEvent* was created. These events are triggered when there is data to be read, written or if the application is being shut down.

Then a thread function was implemented which only purpose is to listen to the *SOCKET* if there is data to be read. This is done using the API function *select()* which is available for Pocket PC 2002. Then when data arrives on the socket the thread function trigger the read event and the *CDuplexWSAThread::Run()* will read the data and reset the event. By doing this, the algorithm for reading and writing data from the Wilma server application could be used here as well.

### 7.4.1 Emulator vs Handheld device

When the work with the wireless application started, an emulator of a device running Pocket PC 2002 was used. This worked well, until the application was moved over to the handheld device that was purchased. What happened was that when data was being prepared to be sent or when data arrived from the server it could get “misaligned” on the handheld device. This was because the emulator was running on the PC that runs an x86 processor while the handheld computer has an WCE ARM processor. So these two processors handle the data differently.

To solve this, the *UNALIGNED* macro was used. This macro is defined in the file windows.h. What had to be done was that everytime the application was doing a cast operation of the form **(data_type*),* it had to be changed to **(UNALIGNED* *data_typ*).*

### 7.4.2 Remaining work

As mentioned before, it is now only possible to connect and login to the server with the handheld client. The communication is working, but if the wireless client receives an alarm it will not be displayed. So there is quite a lot left to do on this application. Things like how the information can be displayed and what information that should be displayed are some of the things which have to be done.

# 8 Discussion

When the work with this thesis started I was prepared to do an imlementation of a wireless alarm receiving client to an existing alarm system. I.e. I really did not think that there would have to be any major work with the existing Wilma server. But as it turned out I have spent most of the time working with the Wilma server, making it communicating using both *Named Pipe* and *SOCKET*. I think there should not be any problems to make it possible to choose between the two in the configuration of the program. Communication using *Named Pipes* is only possible on a LAN (Local Area Network), so this additional implementation of the server now makes it possible to connect a Wilma client to the server over the internet. This will of course need a secure channel such as a VPN.

It has been a new experience to learn how a quite big existing system works and how it is implemented. Of course I have not been looking into everything, because the system is quite large. My supervisor, Thomas Wikström, at Technology Nexus AB here in Umeå did a great job by providing me with the material that was needed and those classes that I had to look into.

## References

[DC01] Michael J. Donahoo and Kenneth L. Calvert, *The pocket guide to tcp/ip sockets: C version*, MK, Morgan Kaufmann Publishers, 2001, ISBN 1-55860-686-6.

[Emi02] Jacqueline Emigh, *Driveby hacking on the go*, Web page, 3 January 2002, <http://networking.earthweb.com/netsecur/article.php/948241>.

[Gei02a] Jim Geier, *802.11 mac layer defined*, Web page, 4 June 2002, <http://www.wi-fiplanet.com/tutorials/article.php/1216351>.

[Gei02b] Jim Geier, *Minimizing 802.11 interference issues*, Web page, 11 January 2002, <http://www.wi-fiplanet.com/tutorials/article.php/953511>.

[Kha01] Mujtaba Khambatti, *Named pipes, sockets and other ipc*, Web page, April 2001, <http://www.public.asu.edu/˜mujtaba/Articles%20and%20Papers/cse532.pdf>.

[Kin01] Coulouris Dollimore Kindberg, *Distributed systems, concepts and design*, Addison Wesley, 2001, ISBN 0-201-61918-0.

[Liu03] James Liu, *How to build a secure wlan*, Web page, 6 February 2003, <http://www.computerworld.com/mobiletopics/mobile/story/-0,10801,78275,00.html>.

[MSD03] Microsoft MSDN, *Named pipes*, Web page, February 2003, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/-ipc/base/named_pipes.asp>.

[Nok03] Nokia, *Questions and answers*, Web page, 2003, <http://www.nokia.se/phones/technologies/javafaq.php>.

[Sch00a] Jochen Schiller, *Mobile communications*, Addison-Wesley, 2000, ISBN 0-780201-398366.

[Sch00b] Bruce Schneier, *Secrets and lies*, John Wiley & Sons, Inc., 2000, ISBN 0-471-25311-1.

[SM03a] Inc. Sun Microsystems, *Cldc specifications*, Web page, 13 May 2003, <http://java.sun.com/products/cldc/>.

[SM03b] Inc. Sun Microsystems, *Java 2 platform, micro edition (j2me)*, Web page, 23 June 2003, <http://java.sun.com/j2me/>.

[SM03c] Inc. Sun Microsystems, *Mobile information device profile (midp)*, Web page, 12 April 2003, <http://java.sun.com/products/midp/>.

[Sym03] Ltd Symbian, *Symbian os - the operating system*, Official Web site, 2003, <http://www.symbian.com>.

[Tec03] Intel Information Technology, *Vpn and wep*, Web page, January 2003, <http://www.intel.com/eBusiness/pdf/it/wp021306.pdf>.

# A Abbreviations

| | |
|---|---|
| AES | Advanced Encryption Standard |
| AP | Access Point |
| API | Application Programming Interface |
| CLDC | Connected Limited Device Configuration |
| CS | Circuit Switched |
| DES | Data Encryption Standard |
| 3DES | Triple-DES |
| DSSS | Direct Sequence Spread Spectrum |
| FDD | Frequency Division Duplex |
| FHSS | Frequency Hopping Spred Spectrum |
| GPRS | General Packet Radio Service |
| GSM | Global System for Mobile communication |
| IEEE | Institute of Electrical and Electronics Engineers |
| ISM | Industrial, Scientific, Medical |
| J2ME | JAVA 2 Micro Edition |
| LAN | Local Area Network |
| LU | Line Unit |
| MAC | Medium Access Control |
| MIDP | Mobile Information Device Profile |
| OH | Overhead |
| OS | Operating System |
| PS | Packet Switched |
| QoS | Quality of Service |
| SSID | Service Set Identifier |
| STA | Mobile Station |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| TDMA | Time Division Muliple Access |
| TKIP | Temporal Key Integrity Protocol |
| VoIP | Voice over IP |
| VPN | Virtual Private Network |
| WAN | Wide Area Network |
| WEP | Wired Equivalent Privacy |
| WLAN | Wireless Local Area Network |
| WPA | Wi-Fi Protected Access |

# B Code

Here you can see in what order the objects are created in the thread functions that are taking care of new clients who are being connected to the server. It is a great deal of code that is missing, but the purpose is not to show exactly how everything is implemented.

- serverPipeThread

```
ConnectNamedPipe(hPipe, ...);
pThread = new CDuplexTransportThread(hPipe,...);
pConsumer = new CNetServer(pThread);
pThread->setConsumer(pConsumer);
pThread->createThread(....);
```

- serverSocketThread

```
nClient = accept(hSocket, ...);
pThread = new CDuplexWSAThread(hSocket,...);
pConsumer = new CNetServer(pThread);
pThread->setConsumer(pConsumer);
pThread->createThread(....);
```