A multiple access protocol for a dynamic car navigation system

van den Eijnden, F.

Award date:
1990

Link to publication
A multiple access protocol for a dynamic car navigation system

Supervisors:
Prof. ir. J. de Stigter (TUE)
Dr. F. Zijderhand (PHILIPS)
Ir. P. Smulders (TUE)

Author:
F. van den Eijnden

May 26th, 1990
Table of contents

Summary

1. Introduction

2. The PREVU system
   2.1. Goals of PREVU
   2.2. Background on GSM
   2.2.1. Synchronization in GSM
   2.2.2. What values are realistic for $t_{path}$ in GSM and ETACS?
   2.3. Description of the graduation subject

3. The multiaccess protocol
   3.1. Specifications of the multiaccess protocol
   3.2. Overview of existing multiaccess protocols
      3.2.1. Fixed TDMA and Fixed FDMA
      3.2.2. Contention systems
      3.2.2.1. ALOHA protocols
      3.2.2.2. Carrier Sense Multiple Access protocols
      3.2.3. Code Division Multiple Access
   3.3. What multiple access protocol should we select?
   3.4. Calculation of the throughput for Reservation ALOHA
   3.5. How do we solve the emergency message problem?
      3.5.1. Giving emergency messages a larger access probability
      3.5.2. Reservation of special emergency timeslots
      3.5.3. Allowing for emergency messages to access in the Reserved timeslots
      3.5.4. Accessing emergency messages with higher power density

4. Interface description
   4.1. The CAROLINE system architecture
   4.2. Functionality of the interface board
   4.3. Interface equipment
      4.3.1. Interface hardware
      4.3.2. Interface software
      4.3.2.1. Working environment
      4.3.2.2. Software implementation
      4.3.2.3. Detailed description of the software implementation

5. Results
   5.1. Results of the implemented protocol
   5.2. Anticipated performance with an increased baudrate
   5.3. Anticipated performance with GSM

CONFIDENTIAL - Internal PHILIPS only
6. **Recommendations**

6.1. Increasing the baudrate 75
6.2. Decreasing the overhead while implementing the protocol 78
6.3. Continuous sending of the complete downlink message 78
6.4. Completion of the Reservation ALOHA protocol 79

7. **Conclusions**

References 81

Appendix A. Circuit diagram of the interface board 86
Appendix B. Program development with AZTEC C80/ROM 87
Appendix C. Calculation of the number of zero-bit insertions 89
Appendix D. Abbreviations 91
Summary

The main aim of the graduation work was to set up a full duplex communication link between a system of mobiles and a base station. There is information flow from base station to traffic center, and vice versa. The channel from the base station to the mobiles is a broadcast channel (point-to-multi-point). This channel provides the mobiles with information about (real time) traffic situations, weather conditions, etc.

Within the mobile there is a static navigation system, named CARIN (CAR Information and Navigation system). With the help of the information from the broadcast channel, CARIN can dynamically plan an optimal route to its destination. The channel from the mobiles to the base station is a multiple access channel (multi-point-to-point). All mobiles within one cell need to transmit their data, such as travelling times, locations, emergency messages, etc., to the base station via this one channel. To perform this task, a multiple access protocol is needed. For now, the communication system makes use of analogue ETACS equipment, but the long term goal is to use digital GSM equipment when it comes available.

A literature study was done to find the most suitable multiple access protocol for the CAROLINE system. A choice was made for an existing multiple access protocol, called Reservation ALOHA, because it answered very well to the CAROLINE specifications. Some new ideas were added to answer specific requirements.

Because a PC, that had a non-DC-free asynchronous RS232 port, was used to process the CAROLINE data, and the signal given to the ETACS transceiver had to be synchronous DC-free, an interface was required. To perform the PC-transceiver interfacing an existing transmitter-receiver interface board was used. The hardware of this interface board is able to handle both an asynchronous NRZ coded RS232 line and a synchronous FM0 coded line. The interface software required to implement the data transfer to perform this conversion was written as part of this study.

An initial software implementation of the chosen protocol has been carried out, and the duplex channels have been operated successfully at 2400 baud. Up to now, only a slotted ALOHA protocol, without acknowledgment, has been implemented.

The downlink requires a timesstructure to enable implementation of the protocol. The overhead to make this timesstructure in the ETACS system is rather small and, in addition, only a few bits in the downlink are needed to control the protocol. Therefore, the maximum implemented downlink data efficiency was 87%. For the GSM system, the timesstructure already exists, and so a better performance can be expected.

The overhead in the implemented uplink was very high. The uplink efficiency was just 54%, for three reasons. The main one being the slow switching times of the transceivers used, the second one being the communication overhead required for the preamble, postamble and CRC of the uplink timeslot within the ETACS system and the last one being the small chosen data length of an uplink timeslot. Again these problems do not exist within GSM and so improved performance can be expected.
Introduction

1. Introduction

A program, called PREVU (PRoduct for Expected Vehicle Usage), has been set up to carry out research into the field of Road Transport Informatics (RTI). This program seeks to find a coordinated solution to the problems of road safety, poor transport efficiency and environmental pollution on a trans-European basis. Within PHILIPS, a testsite named CAROLINE had been set up to investigate several PREVU aspects.

This paper describes a graduation subject done in the CAROLINE team at the PHILIPS Project Centre Geldrop. The paper is a result of a nine month research project carried out for the Technical University Eindhoven.

The graduation work involved an investigation of the most suitable multiple access protocol needed for a dynamic car navigation system, CAROLINE. The CAROLINE system collects and processes detailed, real time information on traffic, road and weather conditions and supplies it to car drivers and authorities. Besides the real time traffic information, predictions are also made for the near future (both short term and long term). The system collects and processes all this information in a traffic center. The traffic center gets its information from authorities, base stations, etc. The base station in turn collects the information from the mobiles. In the car this information is used to advise the driver dynamically on the optimal route to his destination.

The information exchange between mobiles and a base station in CAROLINE uses analog ETACS cellular radio equipment. However the PREVU system will use the pan-European digital cellular mobile radio system (GSM). The graduation work also involved an (initial) implementation of the most suitable multiple access protocol for the ETACS equipment.

In chapter 2 there is a short description of the PREVU system and the CAROLINE testsite. There is also a description of the graduation subject done in the CAROLINE team. In chapter 3 the specifications for the required protocol are described and by means of an overview of existing protocols the most suitable protocol is chosen. Furthermore, the expected throughput performance is calculated and the solution for the emergency messages (which was still not solved) is described.

In chapter 4 there is a more detailed description of the CAROLINE system, with emphasis given to the interface board. In this interface board the multiple access protocol is implemented. In chapter 5, there is a discussion on the results of the initial implementation of the chosen protocol. Finally, in chapter 6 and 7 the required recommendations for the (completion of) implemented multiple access protocol and the conclusions about the chosen (initial implemented) protocol are given.
2. The PREVU System

2.1. Goals of PREVU

In short, the PREVU system is a system which collects and processes detailed, real time information on traffic, road and weather conditions and supplies it to car drivers and authorities. Besides the real-time traffic information, predictions are also made for the near future (both short term and long term). The system uses this information in the car to advise the driver dynamically on the optimal route to his destination.

The main goals of PREVU are:

- to give car drivers (and/or their navigation computers) information on traffic flows, road blocks and local weather conditions, both real-time as well as predictions for the near future;
- to advise the driver on the "best" route based on his own preference, and to provide him continuously with the expected travelling time for his planned route, so providing navigation using dynamic traffic data;
- to collect real-time information from the cars on their positions and their travelling times (anonymously);
- to supply both real-time and long-term statistical traffic information to the authorities;
- to establish a fast digital full-duplex multi-point-to-point communication channel between a set of cars and a base station;
- presenting hazard warnings to drivers, and receiving emergency calls from the cars.

As far as possible PREVU aims to make use of existing navigation equipment (e.g., the navigation system CARIN and automatic vehicle location systems) and the forthcoming pan-European digital cellular mobile radio system (GSM). The attractiveness of the cellular structure [1,2] is that it enables information to be transmitted on anything from a local to a nationwide basis.
The PREVU system

2.2. Background on GSM

It is intended that PREVU will make use of the forthcoming pan-European digital cellular mobile radio system (GSM). GSM has been specified as a narrowband Time Division Multiple Access (TDMA) system operating in the 900 MHz band.

Like all land mobile communication systems, GSM is subject to transmission errors caused by multipath fading. However, it has an additional problem in that the TDMA technique demands synchronization between the BS and MSs in order to avoid message packets from different mobiles overlapping at the base receiver station. This problem is further compounded by the fact that, (a) the mobiles do not know their transmission time-delay (range) from the BS, and (b) their transmission time-delays vary in a random manner. The synchronization problem, and how it is solved within GSM, will be described below.

2.2.1. Synchronization in GSM.

There will be different propagation times between BS and mobiles as a result of the geographical difference in distances and environments between the mobiles and the BS (figure 2.1). Because a MS is moving, the propagation time of its message changes as a function of time. In a TDMA system this will result in a synchronization problem.

This can be explained by means of figure 2.2. At the absolute point of time t₀ the BS broadcasts messages that are time-partitioned, with the length of one timeslot being T. At the absolute point in time t₁, the MS, which is at a certain position A in the cell, receives the broadcast messages. At position A the MS will be receive-synchronized with the BS with a prop-

fig. 2.1 Propagation time differences between different mobiles.
agitation delay $t_{\text{path},1}$.

\[ T = \text{packet time (one timeslot)} \]
\[ = \text{access burst bits, with duration } \tau \]

*fig. 2.2 Propagation delay differences.*

Suppose at the point in time $t_1$ the MS will try to access the channel (the MS has to start an access at the beginning of a timeslot). The access request will be received by the BS at the absolute point of time $t_2$. The time differences $t_2 - t_1$ and $t_1 - t_0$ will be about the same due to reciprocity, so the BS can calculate the value of $t_{\text{path},1} = (t_2 - t_0)/2$. The BS will send this value to the MS, and thus the MS is able to synchronize with the BS, because it now knows the exact (barring a small difference because the MS has moved a little in the meanwhile) beginning of the timeslot at the BS. During the next (small number) of timeslots, the MS can send its packets so that they are received within a timeslot of the BS. However, this will not last forever. It will last until the MS has moved too far from position A, for example position B. The synchronization of the mobiles depends on their position and how fast these positions change.

Suppose the MS will be at position B at time $t_3$. From figure 2.2 one can see that now the
propagation delay is $t_{\text{path},2}$ instead of $t_{\text{path},1}$. Once again the MS does not know the absolute beginning of the BS timeslot. If a MS uses a wrong value of $t_{\text{path}}$, this will lead to interference as the result of collisions between its packets and those from other MSs. To reach synchronization, the procedure described above has to be repeated. This whole process is called adaptive frame alignment.

### 2.2.2. What values are realistic for $t_{\text{path}}$ in GSM and ETACS?

The frequency bands 890 MHz - 915 MHz and 935 MHz - 960 MHz have been allocated for the GSM duplex channels, giving a duplex distance of 45 MHz. GSM consists of 125 frequency bands of 200 kHz each (FDMA, fig. 2.3). Channel number 0 is used as a guard channel, so only 124 channels can be used.

![Frequency Band for GSM](image)

*fig. 2.3 900 MHz frequency band allocated for mobile telephone systems.*

Each 200 kHz frequency band is divided into 8 timeslots (TDMA, fig. 2.4). One user is allowed to use one timeslot. The timeslots are numbered in GSM. The signaling rate in a 200 KHz band is 270.8 Kbit/s. So one bit duration is $1/(270.8 \times 10^3)$ sec = 3.69 μs. One frame is defined as being 156.25 bits. Thus one frame has a duration of 156.25 * 3.69 μs = 0.577 ms.
The PREVU system

1 TDMA frame = 8 timeslots

1 timeslot = 156.25 bit durations (1 bit duration = 3.69 µs)

= guard bits

fig. 2.4 Timeslot assignment and timeslot duration.

Radio signals propagate with the speed of light: \(3 \times 10^8\) m/s. So 1 km distance is covered in 3.3 µs. As shown above, one bit duration is 3.69 µs. Therefore, if a guard period of one bit is inserted into the frame, it will allow a car to travel \(3.69/(3.3 \times 2)\) km \(\approx 550\) m, before requiring resynchronization. The factor 2 has to be added, because the propagation delay is in both directions, the uplink and the downlink.

Within GSM the maximum cellular radius will be about 35 km. This results in a maximum \(t_{\text{path}}\) of \(35 \times 3.3\) µs = 115 µs. Because the delay is in both directions, this value has to be multiplied by 2, resulting in a maximum delay of 231 µs. This is about 62.6 bit durations.

Within GSM two different types of timeslot bursts are defined. The first one is the random access burst, which is used for the initial access, i.e., no synchronization is established yet. The second one is the normal traffic burst, which can be used after synchronization has been established.

In GSM a normal traffic burst has 8.25 guard bits. For the random access burst we will calculate how many guard bits one will need in the worst case. The demand for the random access burst is that the access bits always fall within the known BS timeslot period. This means that the length of the access bits plus twice the maximum \(t_{\text{path}}\) length has to be smaller than the length of a timeslot \(T\) (fig.2.2: \(\tau + 2t_{\text{path}} \leq T\)). As before, in the worst case situation more than 62.6 guard bits are needed. GSM has chosen 68.25 guard bits. This means there are 88 gross random access bits left, but because of fading in the land mobile radio propagation environment, the net number of bits are less than the gross number of bits. The net number of bits is just 8, because GSM has 41 bits for a synchronization sequence, 36 encrypted bits and 3 tail bits.
The PREVU system

Although it is intended that PREVU will be "piggybacked" on GSM, GSM equipment was unavailable for the PREVU test-site (see chapter 4). Therefore, modified ETACS equipment was used instead.

ETACS is a full-duplex Frequency Division Multiple Access (FDMA) system incorporating 30 KHz channels. For the purpose of the PREVU experiments the bands 873 MHz - 873.18 MHz and 918 MHz - 918.18 MHz were exclusively allocated. These bands are again separated by 45 MHz duplex distance, and each band was split into six 30 KHz channels. The signaling rate in these channels is 19.2 Kbit/s, which means a bit duration of about 50 µs. In the PREVU test-site the cellular radius will be 5 km. So the maximum value of 2*t_{path} will be 33 µs, which is even less than one bit duration. So one guard bit will be enough for the random access burst.

The net baudrate within one frequency band however will be less than 19.2 kbit/s. It will range from 10 kbit/s to 15 kbit/s, depending on the chosen encoding technique (e.g., NRZ, Manchester etc.). This reduction of baudrate is introduced, because the transmitted signal has to be DC-free.
The PREVU system

2.3. Description of the graduation subject

In the Project Centre Geldrop a small scale test-site, named CAROLINE, will be setup to investigate several PREVU communication aspects. Figure 2.5 and 2.6 show an overview of the infra-structure part of the PREVU system and of the corresponding in-car part, respectively.

\[\text{fig. 2.5 Infra-structure of the PREVU system.}\]
My graduation subject is concerned with establishing the digital full duplex multi-point-to-point communication channel in the CAROLINE test-site. This full duplex channel will consist of two different one-way channels, called the downlink and the uplink.

The downlink is a broadcast channel from Base Station (BS) to Mobile Station (MS). Most of this broadcast information, for example traffic flows, road blocks, weather conditions etc., is intended to be for all MSs in the cell.

The uplink is intended to be the channel to be used by all MSs to communicate with the BS. The information a MS wants to send to the BS is, for example, the travelling time on a link. The problem which arises here is that several MSs want to simultaneously send their information to the BS (Multiaccess). If two MSs send information at the same time, a collision between the messages sent will arise, and both messages may be destroyed. The first task to be performed is to develop a multiaccess protocol suitable for the specifications within the CAROLINE test-site.
The PREVU system

After finding the most suitable protocol, which will probably be a combination of already existing multiaccess protocols, the task will be to implement this within the framework of the CAROLINE test-environment.

Enumerated, the tasks to be performed will be:
1. Literature study of the most suitable multiaccess protocol(s).
2. Implementation of a time-structure for synchronisation.
3. (Initial) implementation of the chosen multiple access protocol.
The multiaccess protocol

3.1. Specifications of the multiaccess protocol

Before explaining the different multiaccess protocols, we will first give a summarized enumeration of the special demands and restrictions that exist in the CAROLINE test-site:

- An important parameter is the throughput. The throughput can be defined as the percentage of information that will be received without error. Stated mathematically: The throughput can be defined as the product of the offered load and the probability that a packet does not suffer a collision [3]. The BS functions as a transceiver, which passes the collected data from the MSs to a central computer for processing. The more information the central computer gets from all BSs, the better. So also, the more information each BS gets from the different MSs, the better. Therefore, one tries to make the throughput of the access channel as large as possible. For the BS it does not matter from who the received information comes, as the central computer merely needs a statistical average. Sometimes it does not even matter if a message from a MS gets lost. Assuming the message of MS 1 is lost, then it could mean that in the meanwhile MS 2 was occupying the access channel. As a result the BS gets information from MS 2 instead of MS 1. This information has the same priority for the central computer, because it needs only a statistical average.

- The cars that send their information to the BS have to remain anonymous for reasons of privacy, otherwise it will be difficult to sell the system.

- The BS always has to send some controlling information to the MSs. What kind of information this is, will be explained in detail in the next chapters. This controlling information is sent over the downlink. One has to ensure that this control information is as small as possible, because this represents an overhead in the downlink and one wants to send other information on the downlink channel. Therefore, a fully centrally controlled protocol will probably be out of the scope of the CAROLINE test-site.

- An important parameter is the delay [3]. The delay is defined as the time difference between when a message is ready to send and when it is received at the destination. The delay of an emergency message has to be very small. The reliability of an emergency message has to be high, because it always has to arrive at its destination. The delay of the other information messages does not have to be very small. This is in contrast with speech, where delay is an important parameter.

- The access probability within one cell has to be homogeneous. This means that it is not allowed for a MS that is closer to the BS to have a greater probability to access the channel than a MS which is farther away from the BS, because its received signal is stronger. Again, the reason for this restriction is the fact that a homogeneous statistical average is needed. Preferences for receiving messages from vehicles close to the base station will give a wrong traffic density overview.
The multiaccess protocol

- For a multiaccess protocol an important parameter is the stability [3]. There are unstable schemes, which mean that as the offered load increases, these schemes will become unstable, yielding excessive message delay values and diminishing throughput values. For the CAROLINE test-site the stability problem is rather easy to solve. For example one gives an upper bound to the number of retransmissions. This reduction of retransmission will result in an increase of the throughput again to the stable state. A number of messages will be lost, but for the PREVU application this is not important, as long as the lost messages are not emergency messages. As stated previously, the central computer needs a homogeneous average.

- Carrier sense multiaccess schemes will be explained in detail later on. Basically, this means that each station which wants to send information senses the access channel to look if it is free to send. This is not adoptable in land mobile radio, because two terminals can be within range of the central station, but out-of-range with each other. The MSs get no information from each other.

- The number of MSs within a cell will be variable and unknown by the BS. In a later chapter we will try to make an estimation of the maximum number of MSs in one cell. The protocol has to be as independent as possible of the number of users.

- The time at which a MS will generate a message will be random. It depends on what kind of environment the MS is in. If the MS is in a crowded area, for example, a town, it usually has a lot of messages to send, to tell the BS what happened lately. If the MS is in a rural area it will usually have fewer messages to send to the BS.

- One has to try to make the CAROLINE system as representative as possible of GSM.

- The message length, which will be defined later on will depend on the offered traffic information and the protocol we will use.
The multiaccess protocol

3.2. Overview of existing multiaccess protocols

To be able to select a specific multiple access protocol, which is suitable for the CAROLINE test-site, we will first give an overview of existing multiaccess protocols. All of the described protocols [3] are suitable for satellite communication. This, however, has no consequence in the understanding of the basic ideas, which can be used for mobile communication too.

Restrictions.

In the description of the protocols we consider an infinite population of users (terminals) generating new packets according to a Poisson distribution. This is a worst case situation in comparison with a finite number of users. We also suppose that a local source of information generates messages consisting of a variable number of fixed-length data packets.

3.2.1. Fixed Time Division Multiple Access and fixed Frequency Division Multiple Access

In a fixed Time Division Multiple Access (TDMA) system, each user has his own assigned timeslot. The TDMA system is a Fixed-Reservation Access Control scheme (FRAC). If a user wants to send a message, he uses his own assigned timeslot. One frame consists of N timeslots, where N denotes the number of users (Fig. 3.1).

![Frame structure of TDMA](image)

fig. 3.1 Frame structure of TDMA.

An advantage of this multiaccess protocol for satellite is that it is rather easy to implement, especially if the number of users is small. The most important advantage is that the throughput can reach almost 100% if all of the users want to send something. There will never be collisions between sent messages.

However, there are a number of drawbacks. The available bandwidth for one user is just 1/N of the total bandwidth, which will become small for a large number of users. This has different consequences. If a user has a lot of messages to send, the delay will become large. If some users have nothing to send, while other users have a lot of messages to send, the throughput
The multiaccess protocol

will decrease. Moreover, the number of users must be known, and furthermore, the assignment of a fixed timeslot to each user is in contrast with anonymity.

The difference between the Frequency Division Multiple Access (FDMA) protocol and the TDMA protocol, is that each user has his own fixed assigned frequency bandwidth instead of a fixed assigned timeslot. The advantages and disadvantages for FDMA are the same as mentioned before for TDMA.

Summarizing, if each terminal in the network emits a steady flow of messages, so that message inter-arrival times for each terminal have low variance, a fixed scheduling discipline such as TDMA or FDMA will yield an efficient utilization of the channel, as well as low user response times. However, when traffic in a terminal is characterized as bursty, low duty-cycle with a high ratio of peak-to-average traffic density, a fixed-channel assignment is not efficient.

3.2.2. Contention systems

Systems in which multiple users share a common channel in a way that can lead to conflicts are widely known as contention systems. We will start with describing the well known ALOHA systems. Later on we will add some extensions to the pure ALOHA systems to improve some important parameters

3.2.2.1. ALOHA protocols

Pure ALOHA

The basic idea of an ALOHA system is simple [4]: just let the users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding packets will be destroyed. However, due to a feedback property of packet broadcasting, the sender of a packet can find out whether his packet was destroyed or not by merely listening to the downward stream of packets one round-trip time after sending the packet. If the packet was destroyed, the sender just waits a random time and then sends it again. The waiting time must be random, otherwise the same packets will collide over and over.

The idea of the pure ALOHA system is shown in figure 3.2; note that the packets are transmitted at completely arbitrary times.
The multiaccess protocol

The throughput equation for pure ALOHA is [3]: \( S = Ge^{-2G} \), where \( S \) is the throughput and \( G \) is the offered load. This equation is shown in figure 3.3. From this equation it is shown that the maximum throughput of the pure ALOHA protocol is \( 1/2e \), which is \( \approx 18\% \). This can be reached at an offered load of 0.5 attempts per packet time. It has been shown that the delay is small as long as the offered load \( G \) is small. However, as soon as the system begins operating at high offered load values, the delay skyrockets due to collisions, and the throughput diminishes [3].

Fig. 3.2 The pure ALOHA random access channel.

Fig. 3.3 Throughput versus offered traffic for pure ALOHA and slotted ALOHA.
Slotted ALOHA

With the slotted ALOHA method we can double the capacity of the pure ALOHA system. The proposal is to divide time up into discrete intervals, each interval corresponds to one packet. In slotted ALOHA a user is required to wait for the beginning of a slot in order to send his message. The principle of the slotted ALOHA system is shown in figure 3.4.

![Diagram showing slotted ALOHA random access channel](image)

*fig. 3.4 The slotted ALOHA random access channel.*

Waiting until the beginning of a slot means that the vulnerable period is now reduced by half. It can be shown that in slotted ALOHA the throughput equation is: \( S = Ge^{-G} \) \[3\]. Thus, the maximum throughput is \( 1/e \), i.e., \( \equiv 37\% \). This will be reached at an offered load of 1 attempt per packet time. As with pure ALOHA, the delay increases while the throughput decreases at high offered load values \[3\].

Controlled ALOHA

As we have seen above in pure and slotted ALOHA systems, the maximum throughput will be achieved at an optimum offered load \( G \). Of course, the question that arises is how do we find and change this offered load \[5,6\].

The changing of the offered load can easily be done by changing the random waiting time for (re)transmission. If one increases the random waiting time for retransmission, the offered load increases and vice versa.

Let us define \( \alpha \) as being the probability of retransmission in all slots following the original transmission. The mean delay before retransmission is \( 1/\alpha \). Furthermore, the round-trip
The multiaccess protocol

propagation delay will be defined as \( R \) timeslots (in mobile communication, of course, \( R < 1 \), see chapter 3). When the channel is heavily loaded, we want \( \alpha \) to become small. There exist many functions of \( G \) (=offered load) that are suitable for computing \( \alpha \), for example \( \alpha = e^{-G/(R + 1)} \). In reference 3 it is explained why \( \alpha = e^{-G/(R + 1)} \) is chosen.

To estimate the offered load \( G \), we can use the equation of the Poisson distribution [3], which gives the probability that \( K \) packets are generated during a given packet time:

\[
\Pr[K] = \frac{G^K e^{-G}}{K!}
\]

The probability of an empty slot is \( \Pr[0] = e^{-G} \). Taking logarithms, we have \( G = -\ln(\Pr[0]) \). \( \Pr[0] \) can be easily estimated by noting the fraction of slots that were empty during the recent past.

Slotted ALOHA with capture effect

A disadvantage for all of the previously described ALOHA systems is that even the most clever, dynamically adjusted control scheme will never get the throughput above \( 1/e \) (except for small \( N \)). However, there are other methods for making good use of a single shared channel.

In slotted ALOHA it is assumed that all of the received access packets will have the same power. Because of differences in distance, transmission power or shadow fading, the received power of the access packets may differ. Of course one may purposely induce this power difference. If the difference in signal strength of the colliding packets is beyond a certain level, the strongest one may still be received without error, whereas the other(s) fail in accessing the channel. This is called the capture effect [7].

What do we reach by purposely adding this capture effect? We will show the advantage by means of an example. Suppose a user can access the channel with two different allowed power levels, both levels with probability 0.5. For example, power level one is defined by the range from 0 V for a logical '0' to 2 V for a logical '1' (figure 3.5a), while power level two is defined by the range from 0 V for a logical '0' to 5 V for a logical '1' (figure 3.5b).
The multiaccess protocol

In slotted ALOHA, if two users simultaneously access the channel, the packets will collide and both will be destroyed. With the capture effect, both packets will only be destroyed if both users access the channel such that they are received with the same power level. If one user will access the channel with power level one, and the other user will access with power level two, the strongest received one will win (capture effect), and so access the channel without collision. In other words, taking into account the capture effect, a successful access is not only accomplished if one user accesses the channel at a particular time, but also if two, or more, users access the channel at the same time such that they are received with different power levels.

Suppose there are two power levels. Furthermore suppose that the probability of using power level one by a user is the same as using power level two: \( P(\text{Level 1}) = P(\text{Level 2}) = 0.5 \). If two users access the channel at the same time, the probability of success is \( P(\text{success}) = P(\text{Level 1}) \cdot P(\text{Level 2}) + P(\text{Level 1}) \cdot P(\text{Level 2}) = 0.5 \cdot 0.5 + 0.5 \cdot 0.5 = 0.5 \). The probability of no success is \( P(\text{Level 1}) \cdot P(\text{Level 1}) + P(\text{Level 2}) \cdot P(\text{Level 2}) = 0.5 \cdot 0.5 + 0.5 \cdot 0.5 = 0.5 \). The total throughput \( S_{\text{tot}} \) with only this capture effect, i.e. two different power levels will be:

\[
S_{\text{tot}} = S_{\text{slotted ALOHA}} + P(\text{two users access the channel}) \cdot P(\text{success}) \\
= \frac{1}{e} + \left(\frac{1}{2e}\right) \cdot 0.5 \\
\equiv 0.46
\]

One has gained the second factor in the previous equation by adding two different received power levels [7].

By choosing the right levels, it may even be possible to have a successful channel access if three, or more, users access the channel simultaneously. However, this probability will be
very small. It can be shown [7] that this method can result in a maximum total throughput of 53%.

Announced retransmission ALOHA

A new contention-based broadcast multiaccess protocol, called announced retransmission random access (ARRA) [8] requires users to announce the intended location of their potential retransmission slot (to be used in the event of a collision). This protocol increases the capacity of a slotted ALOHA channel by adding a small amount of useful control information, in a special low rate announcement subchannel, to every message transmission. An improved algorithm (called extended ARRA), which involves aborting predicted unsuccessful retransmissions, has a capacity of 0.6.

We would need a low rate announcement subchannel, so that conflicts between new and retransmitted messages can be prevented. This will mean an extra overhead in the downlink of a mobile communication system. However, the ARRA method has a better delay performance in comparison with the previously described ALOHA methods.

In mobile communication there will be a problem of implementing the ARRA protocol. Because the MS is moving, the propagation delay will be variable. This means that the MS will not be properly time-synchronized to the BS. For the MS it will be difficult to announce in which timeslot it will send its retransmission, because it does not know the exact beginning of that timeslot at the moment of retransmission. So, only if the retransmission follows shortly after the transmission itself (see chapter 3), will the MS still be synchronized with the BS, and thus be able to announce the timeslot of the retransmission.

Reservation ALOHA

As we saw before, to reach a good throughput performance, even for high loaded channels, we can make use of TDMA (or FDMA). An ALOHA scheme does not have good throughput performance at high loaded channels. A combination of both protocols can be made [9]. This combination will be called reservation ALOHA. TDMA and reservation ALOHA both have one feature in common: (some) timeslots are reserved for specific stations. With the combination, we will try to make use of the benefits of both protocols, i.e. an ALOHA protocol reaching a good throughput performance, even at high channel loads. There are different reservation methods, depending on the way reservations are made and released. Here we shall briefly enumerate three different reservation schemes.

Method of Binder.

Binder proposed a method that starts with the basic TDMA model and adapts to slotted ALOHA for low channel utilization [10]. As in TDMA, N consecutive slots are grouped together into a frame, with each station "owning" one frame position. If there are more slots than stations, the extra slots are not assigned to anyone. If the owner of a slot does not want it during the current frame, he does nothing. An empty slot is a signal to everyone else
The multiaccess protocol

that the owner has no traffic. During the next frame, the slot becomes available to anyone who wants it, on a contention basis. If the owner wants to retrieve "his" slot, he transmits a packet, thus forcing a collision (if there was other traffic). After a collision, everyone, except the owner, must desist from using the slot. Thus the owner can always begin transmitting within two frame times in the worst case. At low channel utilization the system does not perform as well as normal slotted ALOHA, since after each collision, the collidees must abstain for one frame to see if the owner wants the slot back. Figure 3.6 shows a frame with eight slots, six of which are owned.

![Figure 3.6 Reservation scheme of Binder.](image)

Most of the disadvantages mentioned in the TDMA protocol appear here too, except with Binder's method the throughput at low offered channel load has good performance.

Method of Crowther.

In the method of Crowther, also known as the Reservation-TDMA (R-TDMA) method [11,12], slots do not have permanent owners, but instead, whenever a transmission is successful, the station making the successful transmission is entitled to that slot in the next frame as well. Thus, as long as a station has data to send, it can continue doing so indefinitely. Since it is unlikely that all stations will have long runs of data to send simultaneously, this method works well even when the number of slots per frame is less than the number of stations. In essence, the proposal allows a dynamic mix of slotted ALOHA and TDMA, with the number of slots devoted to each varying with demand. Figure 3.7 shows a frame of eight slots.

The reservation method of Crowther has the advantage that it is applicable even when the number of stations is unknown and varying dynamically. The sending stations can stay anon-
The multiaccess protocol

ymous within this method.

Frame 1

A F C G B D E

Frame 2

A G B D E

Frame 3

A A G D

Frame 4

A A D D

fig. 3.7 Reservation scheme of Crowther.

Method of Roberts.

The method due to Roberts [13], requires stations to make advance requests before transmitting. Each frame contains one special slot, which is divided into \( V \) smaller subslots, used to make reservations (Fig. 3.8).

\[
\text{one frame}
\]

\[
\begin{array}{c}
\text{V subslots for reservation} \\
\text{F data slots}
\end{array}
\]

fig. 3.8 Reservation scheme of Roberts.

When a station wants to send data, it broadcasts a short request packet during one of the reservation subslots. If the reservation is successful, i.e., no collision, then the next regular slot (or slots) is reserved. All users/stations must keep track of the queue length at all times, so that when any station makes a successful reservation it will know how many data
slots to skip before transmitting. Stations need not keep track of who is queued up; they merely need to know how long the queue is. So, in mobile communications, the BS must broadcast this queue length. When the queue length drops to zero, one can do two different things. Firstly, one can revert all slots to reservation subslots, to speed up the reservation process. This is the method that Roberts suggested. Secondly, one can revert all slots to data slots, so one will have slotted ALOHA without an overhead for reservation.

A mixture of both solutions is possible too. This mixture is not exactly the method of Roberts, but it uses his idea. One such example is a reservation structure in which unreserved slots may be used for transmission on a contention basis (Fig. 3.9). This protocol is known as the Interleaved Frame Flush-Out (IFFO) protocol [14].

A mixture of both solutions is possible too. This mixture is not exactly the method of Roberts, but it uses his idea. One such example is a reservation structure in which unreserved slots may be used for transmission on a contention basis (Fig. 3.9). This protocol is known as the Interleaved Frame Flush-Out (IFFO) protocol [14].

![](image)

**fig. 3.9 Frame structure of the IFFO protocol.**

Another combination of the slotted ALOHA protocol and the reservation protocol is known as the Split Reservation Upon Collision protocol (SRUC) [15]. It changes a data slot to reservation subslots only after a collision has been detected.

It can be shown that the throughput will reach almost 100%, theoretically, by choosing the right number of data slots, F, and reservation slots, V, in one frame [16,17,18,19]. In practice, such a high throughput will be impossible, because one cannot make the subslot sizes infinitely small.

*Note.*

In each of the three reservation methods described, the reservations can also be piggybacked. This piggybacking can be used for different applications. Examples are, announcing the last packet of the message, reservation of a next data slot if you have more data to send, and announcing the retransmission datalot in case of a collision. This piggybacking allows for more flexibility in your message packet lengths, and maybe increases the throughput even further.

**Limited-contention protocols**

A TDMA scheme is a collision free protocol, but it has the previously mentioned drawbacks.
The multiaccess protocol

For solving these drawbacks, we will make use of the ALOHA protocols, but these protocols have lots of collisions. Under conditions of light load, contention is preferable due to its low delay. As the load increases, contention becomes increasingly less attractive, because the overhead associated with channel management becomes greater. So, as in all of the previously described ALOHA protocols, we try to limit the number of collisions, so that we will reach a higher throughput, as in the case with the contention-free protocols.

A method will now be described which can be used for all of the previously mentioned contention protocols. It will be obvious that the probability of some station acquiring the channel can be increased only by decreasing the amount of competition. The limited-contention protocols do precisely that.

They first divide the stations up into groups. Only the members of group 0 are permitted to compete for slot 0. If one of them succeeds, it acquires the channel and transmits its packet. If the slot lies fallow, or if there is a collision, they have to restart the process some timeslots later, depending on the number of groups. Similarly, the members of group 1 contend for slot 1. They undergo the same process as the members of group 0, except they do it in other timeslot numbers. By making an appropriate division of stations into groups, the amount of contention for each slot can be reduced.

An example of grouping is the tree algorithm [20]. In this algorithm the tree is searched until there are no competitions in a group (Fig. 3.10).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{tree_algorithm.png}
\caption{The tree algorithm.}
\end{figure}

Group A will firstly try to access the channel. If there are a few collisions, group A will keep competing. If there are a lot of collisions, subgroup B will compete in the next slot. If in subgroup B there are still a lot of collisions, subgroup B shall be divided into smaller subgroups D and E. If there were just a few competitions in subgroup B, subgroup C will be allowed to compete, which, in the case of a lot collisions, will be further subdivided in the next slot, etc..

This subdividing of groups can be easily performed if the number of users, and their identities is known.
3.2.2.2. Carrier Sense Multiple Access (CSMA) protocols

In carrier sense networks all users, who want to transmit some messages, sense the common access channel to see if this channel is free for them to send. This kind of multiple access protocol is especially useful in networks with transmission media like coaxial cable, twisted pairs and fiber optics, because it is easy to sense the access channel in these networks. For the CAROLINE test-site, this kind of multiple access protocol is not appropriate, because two users can be within range of the BS, but out of range of each other. Consequently, they can not sense the common channel.

However, some of these multiaccess protocols will be briefly described, because we may wish to make use of some of their features. We can also adapt these protocols to make them suitable for land mobile radio, by means of central control. This will not really be useful in CAROLINE, but the protocols are mentioned for completeness.

Adaptation of the CSMA to land mobile radio: ICMA

As stated previously, to solve the problem that in land mobile radio communication the terminals cannot sense the carrier transmitted from other terminals, a centrally controlled multiple access system is used. In this system the BS broadcasts idle/busy information about the access channel to all MSs via the downlink [21]. The BS broadcasts busy signals to inhibit other terminals from transmitting a packet. As long as the packet is received, the BS keeps transmitting the busy signal. This protocol is called Idle-signal Casting Multiple Access (ICMA).

A problem with ICMA arises because of the delay, D (Fig. 3.11), which is the time delay between the start of packet transmission at a terminal, and its detection at another terminal by way of the BS. A collision of packets can occur when terminals attempt to transmit a packet within D, after another terminal has already started a packet transmission.
The multiaccess protocol

![Diagram of the multiaccess protocol](image)

<table>
<thead>
<tr>
<th>upward</th>
<th>Pre</th>
<th>Word 1</th>
<th>Word 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>downward</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>

- **Pre** = Preamble signal  
- **B** = busy signal  
- **D** = Packet detection delay  
- **I** = Idle signal

**fig. 3.11 ICMA transmission scheme.**

This ICMA protocol is especially useful for terminals that want to send long message bursts, so that the **D** is relatively small compared to the packet size. It can be seen that **D** is relatively small in mobile radio communication compared to the propagation delay in satellite communication. So the delay problem is not so severe for land mobile radio communications.

A big drawback of the ICMA protocol is the overhead of 100% one has to expend on the downlink. One needs a complete new channel for this protocol.

**The nonpersistent CSMA**

The idea here is to limit the interference among packets by always rescheduling a packet which finds the channel busy upon arrival. More precisely, a ready terminal senses the channel and operates as follows:

1) If the channel is sensed idle, it transmits the packet.
2) If the channel is sensed busy, then the terminal schedules the retransmission of the packet to some later time according to the retransmission delay distribution. At this new point in time, it senses the channel and repeats the algorithm described.

A slotted version of the nonpersistent CSMA can be considered in which the time axis is slotted and the slot size is \( \tau \) seconds. All terminals are synchronized and are forced to start transmission only at the beginning of a slot. When a packet's arrival occurs during a slot, the terminal senses the channel at the beginning of the next slot and operates according to the protocol described above.
The multiaccess protocol

It can be shown [22] that the throughput of this protocol easily exceeds 90%, but will never reach 100%, because there will still be collisions of data packets. If one senses the channel to be idle, this does not necessarily mean that the channel is idle at that moment, due to the propagation delay of a message sent out by another user.

The \( p \)-persistent CSMA

This is basically the same idea as nonpersistent CSMA, with the differences that a message will be sent with probability \( p \) if the channel is sensed idle, and the station persistently senses the state of the channel. More precisely, a ready terminal senses the channel and operates as follows:

1) If the channel is sensed idle, it transmits the packet with probability \( p \).
2) If the channel is sensed busy, it \textit{waits} until the channel goes idle (i.e. persisting on transmitting) and only then transmits the packet with probability \( p \).

In the case of \( p=1 \), we note that, whenever two, or more, terminals become ready during a transmission, they wait for the channel to become idle (at the end of that transmission period), and then they transmit simultaneously with probability one. A conflict will always occur with probability one. The idea of randomizing the starting time of a transmission period suggests itself for interference reduction and throughput improvement.

It can be shown [30] that with a small value of \( p \) the throughput of the \( p \)-persistent CSMA can exceed the nonpersistent CSMA, but because of the small value of \( p \) the delay will be increased.

CSMA with Collision Detection (CD) or with Collision Avoidance (CA)

With the CSMA-CD scheme, in addition to detecting if the access channel is busy or idle, one also detects if there is a collision on the access channel. If there is a collision, one immediately stops transmitting the message. In ICMA-CD the BS that detects collision among several upward packets will broadcast stop signals. Since this broadcasting of stop signals aborts colliding transmissions, and shortens the wasted transmission period of the upward channel, ICMA-CD can improve on the upward channel efficiency.

The CSMA-CA is a variant of the CSMA-CD technique. It utilizes all of the best features of the CSMA-CD, but resolves contention before any data is transmitted rather than by detecting colliding packets. This avoidance is done by means of accessing the channel with just a small part of the message (or for example a kind of synchronization character) instead of the total message. We would expect a throughput improvement with this collision avoidance method. However, reference 23 shows that there is no throughput improvement compared to CSMA-CD. It is shown in reference 23 that the CSMA-CA gives other desirable characteristics, such as a reduction in contention. This can be important for emergency messages.
3.2.3. Code Division Multiple Access (CDMA)

A multiplexing scheme that has small delays, independent of the offered load, and that gracefully handles collisions on a common channel, is Code Division Multiple Access (CDMA). CDMA is the same as Spread Spectrum Multiple Access (SSMA) [24]. This scheme allows for transmission on the same frequency, and at the same time, by distinguishing the multiple simultaneous transmissions based on a unique pattern used by each sender.

A spread spectrum signal is sent as a series of symbols spread across a wide frequency range. By spreading the power of the signal across the spectrum, the power density at any single frequency in the range is very low. Figure 3.12 shows what happens with an AM signal if it is handled with spread spectrum technology.

\[ \text{fig. 3.12 Spread spectrum.} \]

CDMA makes interception unlikely, because it can work in the presence of a poor signal to noise ratio. A long series of symbols to represent a single piece of data is used to combat interference. As long as the received signal is a series close to one of several expected series, the transmitted data can be correctly identified by the receiver.

So in CDMA, the multiplexing is performed based on codes. Each user has a unique code. The codes of the different users have to be mutually orthogonal in order to decrease their cross-correlation coefficients.

The gain of multiplexing based on codes, is that it allows transmission without setup time, since there is no need to coordinate when multiple stations may transmit, and handles contention as an integral part of the multiplexing scheme. If the BS is "intelligent", and fast enough, the throughput can reach up to 95%.
The multiaccess protocol

However, there are a lot of drawbacks. It is clear that the codes have to be very large if there are a lot of users, because of the previously mentioned demand for orthogonality. An increase in the code length usually entails a concomitant increase in the code bit-rate, and hence results in a larger bandwidth. Furthermore, the base station has to know the codes of the different users. This means the users cannot be anonymous. The CDMA protocol is rather difficult to implement, especially because of the high data rate, which will make the receivers rather complex.
The multiaccess protocol

3.3. What multiple access protocol should we select?

The most important factor for the CAROLINE test-site is trying to attain a high throughput. From table 3.1, the first choice would be the TDMA/FDMA method. However, this method is unsuitable for our application, because the BS knows neither the identities of the MSs in its area, nor how many MSs there are. Therefore, there is an anonymity problem.

Because of the high throughput demand, the second choice would be the CSMA protocol, but from table 3.1, it can be seen that in land mobile communication one can not make use of the CSMA protocol. This is because two users can be within range of the BS, but out of range of each other. Instead of CSMA, one can make use of the ICMA protocol, but with this protocol arises the problem of too large an overhead, which is not allowed in the CAROLINE test-site.

To satisfy the throughput demand, the next choice will be the reservation ALOHA protocol. Fortunately, there is at least one solution for each of the drawbacks of this protocol.

The problem of anonymity will be resolved by the protocol chosen by GSM. Stated briefly, a MS that reserves a timeslot knows in which absolute timeslot it has made its reservation. Then, if the BS broadcasts that a reservation has been made, a MS knows it is its reservation, because if there was a collision at the moment of its reservation, a reservation would not be made. A MS does not need to identify itself, because the moment of reservation is unique. The reservation identification is done by means of timeslot number instead of vehicle number.

However, there is a reason why a MS should identify itself in some (anonymous) way. Suppose two MSs try to make a reservation in the same absolute timeslot, normally in this case there will be a collision. However, it can happen that the signal of MS 1 will be stronger then the signal of MS 2. If this difference is rather large the BS will just receive the strongest signal. It will make a reservation for a MS and broadcast this reservation. The problem arises that both MSs will think it is their reservation. One (=the right MS) of the two MSs will receive the right value of $t_{path}$, but the other one (=the wrong MS) will get a wrong value. The wrong MS will start sending messages that are completely out of synchronization. These packets will interfere with the correctly synchronized packets of the right MS. The result is that all packets, both of the right MS and the wrong MS, can be destroyed, and furthermore, GSM will also be perturbed. This problem can be solved adevely by means of giving a random number (the users are still anonymous) to each new message that a MS uses to try to reserve a timeslot. This random number can be used by the BS to give an acknowledgment to MSs about which timeslot is reserved for whom. The probability that two MSs will attempt to access the channel within the same timeslot, and with the same random number, will be very small (the product of two independent small probabilities). However, the random number will increase the overhead in the downlink.

The second problem of a possible large delay for emergency calls is less than it appears to be. The question is: "What do we call a large delay" ? The values presented in table 3.1 are valid for satellite speech communication. In speech a delay of a few seconds is a problem, as
Table 3.1 Overview of multi-access protocols studied.

<table>
<thead>
<tr>
<th></th>
<th>TDMA/FDMA</th>
<th>Slotted ALOHA</th>
<th>Announced retrans. ALOHA</th>
<th>Reservation ALOHA</th>
<th>CSMA</th>
<th>ICMA</th>
<th>ALOHA with capture effect</th>
<th>CDMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay with small offered load</td>
<td>large</td>
<td>very small</td>
<td>very small</td>
<td>small</td>
<td>very small</td>
<td>very small</td>
<td>very small</td>
<td>very small</td>
</tr>
<tr>
<td>Delay with big offered load</td>
<td>small</td>
<td>large</td>
<td>large</td>
<td>large</td>
<td>small</td>
<td>small</td>
<td>medium</td>
<td>very small</td>
</tr>
<tr>
<td>Estimation of the max. throughput</td>
<td>99%</td>
<td>37%</td>
<td>60%</td>
<td>80%</td>
<td>90%</td>
<td>90%</td>
<td>53%</td>
<td>95%</td>
</tr>
<tr>
<td>Suitable for large number of users</td>
<td>difficult</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Anonymity</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Overhead in downlink</td>
<td>small</td>
<td>small</td>
<td>small</td>
<td>medium</td>
<td>small</td>
<td>large</td>
<td>small</td>
<td>small</td>
</tr>
<tr>
<td>Overhead in uplink</td>
<td>very small</td>
<td>very small</td>
<td>medium</td>
<td>small</td>
<td>small</td>
<td>small</td>
<td>small</td>
<td>very high</td>
</tr>
<tr>
<td>Easy to guarantee stability</td>
<td>very easy</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>very easy</td>
<td>very easy</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Suitable for land mobile commun.</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>small</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Delay on emergency calls</td>
<td>small</td>
<td>large</td>
<td>large</td>
<td>large</td>
<td>large</td>
<td>small</td>
<td>large</td>
<td>very small</td>
</tr>
</tbody>
</table>
The multiaccess protocol

this destroys the speech quality. For emergency messages, a delay of a few seconds will be allowable, but longer delays maybe unacceptable. An example of a realization that avoids this delay problem is one that makes use of special reserved emergency only timeslots. This solution looks like a TDMA scheme, with the difference that a slot is not reserved for a particular terminal, but for a particular type of message, namely emergency messages.

To reduce the delay even further, one could make the probability of sending an emergency message much higher than the probability of sending any other type of message. If collisions occur in the specially reserved emergency slots, the BS reserves more slots for emergency messages, and broadcasts these slot numbers to all MSs. The more of these specially reserved emergency time slots, the less the probability of collision.

The remaining protocols not discussed yet are the announced retransmission ALOHA, ALOHA with capture effect and the slotted ALOHA. They have the drawback of smaller throughput, but, we can of course make use of the ideas proposed in these protocols.

The idea of controlling the protocol will be applied in the Reservation ALOHA scheme to solve the stability problem for which there are three different solutions. For each of the three solutions there is one goal, namely to reduce the offered load.

The first method is to reduce the retransmission rate in case of too many collisions. By increasing the random retransmission time, the probability of a collision at a later moment will be reduced. An increase in random retransmission time will mean a larger standard deviation and so smaller collision probabilities. By means of this method the standard deviation can be changed continuously.

The second method entails dividing up the participating users into groups after collisions have been detected. Since the participating group is now smaller, so the probability of a collision will also be smaller. The dividing can be done easily by means of random access numbers, for example, grouping into even and odd random access numbers, etc. The result might be expected to be the same as in the first method, namely an increase in the standard deviation. However, in contrast to the first method, with this method the standard deviation can only be changed in discrete steps.

The third method will be very easy to implement in the CAROLINE test-site, whereby one just limits the possible number of retransmissions. If there are too many collisions, the upper bound of retransmission times will be decreased. If a MS reaches the upper bound it simply drops its packets.

The third method will not be suitable as a stand alone method, because it has a fixed average for the random retransmission time. Therefore, it is not very flexible. For example, if one chooses a large average, which means a low retransmission probability, this will result in a bad throughput performance if the offered load is small. Conversely, if one chooses a small average, a bad throughput performance results if the offered load is high.

The three methods can also be easily used the other way around, i.e., decreasing the stan-
The multiaccess protocol

dard deviation of retransmission probability, or increasing the upper bound of the number of retransmissions, when there are a large number of idle timeslots in the uplink. This controlling policy will result in a stable protocol whilst reaching the best possible offered load.

It will be difficult to use the retransmission announcing policy. There are two reasons for this. Firstly, the propagation delay of the message will change in time, as described in chapter 3. In order to avoid synchronization problems between BS and MS, one cannot make the time between an announcement and a (re)transmission too large. In chapter 3 one saw that this is not such a big problem. For land mobile radio, the time before losing synchronization is rather large, depending on the number of guard bits. As before, each guard bit will mean a travelling distance by the MS of more than 500 metres. Secondly, there is the drawback of increasing the overhead. The announced retransmission ALOHA protocol is based on the assumption that there is a special retransmission announcement timeslot. Because we are restricted to the minimum timeslot size, T, of GSM, which is rather large, this protocol will result in a sizeable overhead.

The ALOHA system with capture effect can be accomplished in the reservation ALOHA scheme. However, in land mobile communication, the capture effect already exists due to the differences in received power levels from MSs having different ranges from the BS. The need for varying the MS transmit-power to induce a more pronounced capture effect, will depend upon the magnitude of the existing power-level differences.

There will be another reason why we should purposely have different power levels. The access packets of a MS close to the BS will generally have a higher received signal strength than the packets of a MS further away from the BS. This is in contrast to the restriction that the probability of accessing the channel has to be homogeneously distributed across the cell. This problem can be resolved by means of allowing different access signal strengths for each MS, e.g., assigning different access signal strengths depending on the MS position. A MS further away from the BS will be allowed to transmit with a higher signal power than a MS close to the BS. This would also require a detailed knowledge of the BS-position in order that the line of sight distance from BS to MS can be calculated. This problem, however, is not important in the first instance.
3.4. Calculation of the throughput for Reservation ALOHA

As described before, the reservations in the Reservation ALOHA system are made by means of the slotted ALOHA random access protocol. In slotted ALOHA the throughput is rather easy to calculate. Reference 3 shows the throughput, S, is given by: \( S = Ge^{-G} \), with \( G \) the offered load. The maximum throughput will be reached with \( G = 1 \) attempt per timeslot, and will be \( 1/e \), which is about 37%. Reference 3 also shows that if the system is operating at \( G = 1 \), the probability of an empty slot is about 37%. So the best we can hope for, using slotted ALOHA, is 37% of the slots empty, 37% successes, and 26% collisions (two or more accesses in the same timeslot).

The uplink access channel with the Reservation ALOHA system will look like figure 3.13.

In the continuous timeslot stream there will be alternating random access timeslots (indicated as A in fig. 3.13), in which collisions might happen, and reserved timeslots (indicated as R in fig. 3.13), in which it is assumed here for the purpose of calculation that there will never be a collision.

With the assumption, that in the R timeslots collisions will never happen, their throughput is 100%. For the A timeslots, it is known they satisfy the slotted ALOHA performance values, i.e., a throughput of about 37%. Suppose, for the sake of the calculation, one filters out the R timeslots in the continuous uplink data stream (fig. 3.13). The resulting abstracted data stream consists of only A timeslots.

Consider the following example. Suppose this newly created data stream (the A timeslots) consists of 100 timeslots. Furthermore, suppose one reserves two timeslots after each successful access. It will mean that the original data stream consisted of \( 100 + 2 \times 37 = 174 \)
The multiaccess protocol

timeslots.

The throughput is normally defined as the fraction of successfully transmitted packets in the total datastream. In our example this will be \((3 \times 37) / 174\), which is about 64%. One can do these calculations with different message lengths. When we define the following variables:

\[ \alpha = \text{probability of success}, \]
\[ \beta = \text{average number of reserved timeslots per message}, \]
\[ \beta + 1 = \text{message length in timeslots}, \]
\[ \gamma = \text{number of Access timeslots (see fig. 3.13)}, \]

we will have the following general equation for the "gross" throughput:

\[
\text{"gross" throughput} = \frac{(\beta + 1) \alpha \gamma}{\gamma + \beta \alpha \gamma} = \frac{(\beta + 1) \alpha}{1 + \beta \alpha} = \frac{\beta + 1}{\beta + 1/\alpha}
\]

Fortunately, this equation is independent of \(\gamma\). Remember \(\alpha < 1\), so \(1/\alpha > 1\), which means the throughput can never reach 1, i.e., 100%.

The results of these calculations are shown in table 3.2. If one uses, for example, only half a timeslot, one is wasting the other half of that timeslot. That is why it is advisable to make the message length an integer number of timeslots in order to reach a high throughput.

<table>
<thead>
<tr>
<th>message length in timeslots ((\beta+1))</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentage throughput</td>
<td>37</td>
<td>54</td>
<td>64</td>
<td>70</td>
<td>74</td>
<td>85</td>
<td>92</td>
</tr>
</tbody>
</table>

However, one is not interested in this defined "gross" throughput. When using the Reservation ALOHA protocol within the GSM system, there is an overhead, which diminishes the throughput. We will mention two important reasons of this extra overhead: (1) for the GSM system, and (2) for the ETACS system.

(1) There is a large overhead because of the synchronization problem within GSM as described in chapter 3. The synchronization problem arises due to the restriction that the first packet has to be an access packet, which is not part of the message itself. So one complete timeslot is used for making the reservation. This overhead will result in a decreased throughput, which we will call the net throughput \(GSM\).

The net throughput \(GSM\) in the above example is actually less than 64%, being \((2 \times 37) / 174\),
The multiaccess protocol

which is about 42%. One can make the same calculations with different message lengths. When using the previously defined variables, the general equation for the net throughput \(GSM\) is:

\[
\text{net throughput}_{GSM} = \frac{\beta \alpha \gamma}{\gamma + \beta \alpha \gamma} = \frac{\beta}{\beta + 1/\alpha}
\]

The results of these calculations are shown in table 3.3.

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
\text{message length in timeslots (\(\beta + 1\))} & 1 & 2 & 3 & 4 & 5 & 10 & 20 \\
\hline
\text{percentage throughput} & 27 & 42 & 53 & 60 & 65 & 79 & 88 \\
\hline
\end{array}
\]

Table 5.3. Net throughput\(_{GSM}\) calculations depending on message length. \((\alpha = 0.37)\)

(2) There is a large overhead because of the fact that there has to be a timestructure within the Reservation ALOHA scheme, which is an expansion of slotted ALOHA (see section 3.2.2.1). A timestructure will mean, there has to be synchronization (sync) patterns between the packets of data. The overhead due to the sync patterns will depend on the length of the data packets in between the sync patterns. The diminished throughput due to the timestructure, we will call the net throughput\(_{timestruc}\).

Suppose one uses a sync pattern of 8 bits, i.e., one octet. Furthermore, suppose one timeslot consists of 100 bits, and suppose one reserves two timeslots after each successful access. In this example, the net throughput will be \((100 / 108) \times ((3 \times 37) / 174)\), a factor \((100 / 108)\) less than the "gross" throughput. We can do the same kind of calculations with different timeslot lengths. When defining the extra following variables:

\(\delta = \text{timeslot length,}\)

\(\varepsilon = \text{sync pattern length,}\)

we have the following general equation for the net throughput\(_{timestruc}\):

\[
\text{net throughput}_{timestruc} = \frac{\delta}{\delta + \varepsilon} \times \frac{(\beta + 1)}{\beta + 1/\alpha}
\]

The results of these calculations are shown in table 3.4, where we suppose a message length of 3 timeslots and a sync pattern of one octet. It will be clear that the throughput in table 5.4 will reach up to 64% (see table 5.2).
The multiaccess protocol

Table 5.4. Net throughput \( \text{timesruct} \) calculations depending on timeslot length.
\( (\alpha = 0.37, \beta + 1 = 3, \varepsilon = 8) \)

<table>
<thead>
<tr>
<th>timeslot length in bits (( \delta ))</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentage throughput</td>
<td>48</td>
<td>55</td>
<td>57</td>
<td>59</td>
<td>61</td>
<td>62</td>
<td>63</td>
</tr>
</tbody>
</table>

Within GSM one timeslot consists of about 59 databits. For ETACS we will use 64 databits, because it is an integer number of octets, which will simplify the software in the CAROLINE test-site, since we are using an 8 bit micro-processor (i.e., Z80). With 64 databits we are nearly compatible with GSM. Using 64 databits with an eight bit sync character will mean a total decrease of the throughput by a factor \( 64 / 72 = 8 / 9 \approx 0.89 \). These calculations are shown in table 5.5.

Table 5.5. Net throughput \( \text{timesruct} \) calculations depending on message length.
\( (\alpha = 0.37, \delta = 64, \varepsilon = 8) \)

<table>
<thead>
<tr>
<th>message length in timeslots (( \beta + 1 ))</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentage throughput</td>
<td>33</td>
<td>48</td>
<td>57</td>
<td>62</td>
<td>66</td>
<td>76</td>
<td>82</td>
</tr>
</tbody>
</table>

Taking into account both the throughput decrease due to the GSM synchronization problem, which is much larger than within the CAROLINE test-site, where we are using ETACS (see chapter 3), and the throughput decrease due to the timesstructure, which already exists within GSM (see chapter 3), we will call the net throughput \( \text{final} \). Table 5.6 shows theoretical values of this throughput. These values are not realistic within the CAROLINE test-site, because of the small synchronization problem that exists (see chapter 3).
The multiaccess protocol

Table 5.6. Net throughput\textsubscript{\text{final}} calculations depending on message length.  
(\(\alpha = 0.37, \delta = 64, \epsilon = 8\))

<table>
<thead>
<tr>
<th>message length in timeslots ((\beta + 1))</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentage throughput</td>
<td>24</td>
<td>37</td>
<td>47</td>
<td>53</td>
<td>58</td>
<td>70</td>
<td>78</td>
</tr>
</tbody>
</table>

More realistic for the CAROLINE test-site are the throughput values shown in table 5.5, i.e., the "gross" throughput, taking into account the overhead due to implementing a time-structure within the ETACS system. However, more interesting, of course, are the throughput values shown in table 5.3. Because later on, when we are using the GSM system, we do not have overhead due to the implementation of a time-structure. The values of table 5.3 are shown in fig. 5.14.

![Graph showing net throughput depending on message length](image)

\textit{fig. 5.14 Net throughput}\textsubscript{GSM} depending on message length.

From figure 5.14 and its associated equation: net throughput\textsubscript{GSM} = \(\beta / (1/\alpha + \beta)\), it is clear that the net throughput\textsubscript{GSM} can reach up to 100\%, when choosing \(\beta\) very large. However,
The multiaccess protocol

the relative improvement we gain with increasing the message length, quickly decreases. For example, gaining a throughput increase of a factor $60/53 \approx 1.13$, will mean needing a larger message length of a factor $4/3 \approx 1.33$, while gaining a throughput increase of a factor $88/79 \approx 1.11$ will mean needing a larger message length of a factor $20/10 = 2$. The relatively small throughput improvement with large message lengths is due to the overhead for making a reservation. One cannot divide a timeslot into smaller subslots (minislots) for making the reservation. So one needs one complete (big) timeslot for making the reservation.
3.5. How do we solve the emergency message problem?

In paragraph 3.1 we say that the delay for the "normal" messages does not have to be very small. However (a), the delay of an emergency message has to be very small. Furthermore (b), the probability that an emergency message reaches its destination has to be one. Said in another way: an emergency message is never allowed to get lost. Demand (b) can be easily performed by sending out the emergency message continuously until there is an acknowledgement. Demand (a) is more difficult to solve. This demand involves the fact that we have to recognize different types of messages. First, we will enumerate four different possible solutions for solving the problem. In the next subparagraphs, a detailed description of the different solutions will be given. The four different solutions are:

1. emergency messages are given a larger probability to access in the Access timeslots (fig. 3.13) than normal messages,
2. reservation of special emergency timeslots,
3. allowing for emergency messages to also access in the Reserved timeslots (fig. 3.13),
4. allowing emergency messages to access with a higher power density.

The choice of the solution will also depend on its complexity, because the probability of having an emergency message is not large.

3.5.1. Giving emergency messages a larger access probability

What will happen with the delay when allowing emergency messages to access with a higher probability is difficult to answer. A contradiction appears to arise, such that on the one hand, we expect a delay decrease, while on the other, we expect a delay increase, and we will now try to explain this.

(1) To explain why we expect a delay decrease with the proposed solution is done with the help of fig. 3.13. As stated previously, for the R timeslots it is known that they are all occupied. For the A timeslots, it is known that they satisfy the slotted ALOHA performance values, i.e., 37% of these timeslots are idle. Firstly, we give an explanation by means of an example. Suppose for the sake of calculation, the data stream of A timeslots consists of 100 timeslots, that means 37 timeslots are idle. Furthermore, suppose we have message lengths of 2 timeslots. As described in the previous paragraph, one will need at least 3 timeslots for sending this message within GSM (the first one is just used for making the reservation). Backwards calculation shows that we started with a total number of 100 (= A timeslots) + 2 * 37 (= R timeslots) = 174 timeslots, if the message length is 2 timeslots. The percentage of idle timeslots will be 37 / 174, which is about 21%. One can do the same type of calculation for different message lengths. When we define the variables $\alpha$, $\beta$, $\gamma$ to be the same as in paragraph 3.4, we will have the following equation for the percentage of idle timeslots:

$$\text{fraction of idle timeslots} = \frac{\alpha \gamma}{\gamma + \beta \alpha \gamma} = \frac{1}{\beta + 1/\alpha}$$

The results are shown in table 3.7.
The multiaccess protocol

Table 3.7. Idle timeslot calculations depending on message length.
\((\alpha = 0.37, \delta = 64, \epsilon = 8)\)

<table>
<thead>
<tr>
<th>message length in timeslots ((\beta+1))</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentage idle timeslots</td>
<td>27</td>
<td>21</td>
<td>18</td>
<td>15</td>
<td>13</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

As said before, in the PREVU test-site 1 bit duration is 50 \(\mu\)s. One frame probably consists of 64 bits. This means one frame has a duration of 3.2 ms, and so there are 312.5 frames/sec. For example, for a message length of 3 timeslots, this will mean there are about 56 idle timeslots per second. One should expect that, by means of increasing the probability of sending an emergency message, i.e. decreasing the (re)transmission delay, one should easily access the channel with a small delay, because of the presence of so many idle timeslots.

(2) To explain why we expect a delay increase with the proposed solution is done with the help of fig. 3.15 and fig. 3.16 [3].

\[\text{Delay} \quad (\text{slots})\]

\[1\quad 2\]

\[\text{G (attempts per packet time)}\]

\[\text{fig. 3.15 Delay as a function of the offered load for slotted ALOHA.}\]
When allowing emergency messages to access with a larger probability in the Access timeslots (fig.3.13), we expect to diminish the throughput, because we are not working at the maximum offered load $G = 1$ attempt per packet time in the slotted ALOHA scheme. Increasing the access probability will mean increasing the offered load and so increasing the delay (fig. 3.14). So with the proposed solution, we will both decrease the throughput and increase the delay, so decreasing the performance (fig.3.15).

*How can this contradiction be solved?*

This contradiction is introduced because we did not look to the number of MSs that want to send out an emergency message simultaneously. If there are just a few MSs that want to send out an emergency message, the first case, i.e., an expected delay decrease, will be true. If almost all MSs in a cell want to send out an emergency message, the second case, i.e., an expected delay increase, will be true. It is expected that there will be a maximum number of cars allowed to send emergency messages simultaneously, so that with the proposed solution of increasing the probability of accessing the channel, there will be a delay decrease.

Exactly forecasting what will happen with this proposed solution is rather difficult, because there will be a system with two different user groups, each with its own access probability. It is recommended to investigate the expected maximum number of MSs that are allowed to send out an emergency message simultaneously, while still achieving a delay decrease. However, what we can predict already is that, when there are a lot of MSs sending emergency messages simultaneously, the proposed solution is not very useful.
3.5.2. Reservation of special emergency timeslots

The largest profit we gain by making reservations of special emergency timeslots is that one can recognize the type of message. If there happen to be a lot of collisions in these emergency timeslots, the BS can decide to reserve more timeslots for emergency messages. The exact time before the BS will make this decision is for further investigation. However, it is expected that setting up this extra allocation of reservations for emergency messages is rather difficult to implement, because it needs a very fast feedback (see paragraph 3.2.2.1, controlled ALOHA).

The question that arises out of this solution is: "How many specially reserved timeslots (per second) do we need for emergency messages" ? The easiest way to give an answer to this question, is done by means of a numerical example. As calculated previously, for example, when using a message length of 3 timeslots, the percentage of idle timeslots is 18%, which will mean 56 idle timeslots per second in the CAROLINE test-site (paragraph 3.5.1). Reserving one extra timeslot per second, which will mean an extra overhead of $\frac{1}{312.5} \approx 0.3\%$ (paragraph 3.5.1), does not help at all. Not only because there are already 56 idle timeslots per second, but also, which is more important, even if the access in the emergency timeslot succeeds, with a high access probability for emergency messages of $p \leq 1/10$, the delay normally will be more than 10 seconds. The delay will be even larger, because there can be collisions or normal error bursts in the (emergency) timeslots. So we will need much more than one emergency timeslot per second. The drawback of this will be the extra overhead we need in the downlink. Because emergency messages does not happen very often, the solution looks to complex.

3.5.3. Allowing for emergency messages to access in the Reserved timeslots

If an emergency message is allowed to also access in the Reserved timeslots (fig. 3.13), data in these timeslots will be destroyed because of collision. The BS does not know that the destruction of the data is accomplished by a collision with an emergency message, or by a normal error burst. However, if there will be a lot of destroyed data in the Reserved timeslots, the BS can decide to reserve more timeslots for emergency messages. It is recommended to investigate the exact time before the BS will make this decision.

There is a solution in which we do not need to know this waiting time. In the description of the Reservation ALOHA scheme so far, we have not mentioned how many timeslots for one message will be reserved by a BS at the same time.

Of course, if a message length is 3 timeslots, for example after the first successfully random access, which is used for making the reservation in GSM (see chapter 3), the BS can decide to reserve timeslot number x, y and z immediately. However, the drawback which is introduced by means of reservation of 3 timeslots at once, is the non-flexibility. Another solution for making the reservations is by means of making one reservation at once. So after the first successful random access the BS will reserve only one timeslot. If the MS will need more than one timeslot for transmitting its message, it can announce in its reserved timeslot it has more data to transmit (kind of piggybacking), so another timeslot will be reserved. This pro-
cess can be repeated until the MS's complete message is sent. When using the described reservation method, the profit is that an automatically anonymous acknowledge method for the complete message is accomplished. Assignment of a reservation will mean an acknowledge (see paragraph 3.3). From the moment an acknowledge has been given, this will mean a point-to-point connection has been established between the BS and the MS. If a packet in the uplink is destroyed, the BS is able to ask the MS to re-send that packet because of the established point-to-point connection. However, this handshake principle will make the protocol more complex, and thus, is rather difficult to implement. Furthermore, we do not need it, because it is allowable to lose a normal message (see paragraph 3.1).

So, the proposal will be, not to implement this handshake idea, but just quit the connection that has been established so far, i.e., reserve no more timeslots.

The drawback of this method is, that from the moment that the data is destroyed in one timeslot, the complete message will be destroyed, because a new timeslot is not reserved for completing the message (no acknowledge). If the BS will receive two "half" messages later on, it will not be able to reconstruct these into a complete message, because the message is anonymous (no message identity). This also means that the message length must not be too large, because destruction of one timeslot, will mean destruction of the total message. This demand is in contrast to the idea of Reservation ALOHA, in which the message length has to be large in order to reach a high throughput (paragraph 3.2.2.1). If one has a reliable channel, it is allowable to have a large message length.

However, the mentioned drawback will be a gain for the implementation of emergency messages. Destroying the data in the Reserved timeslots purposely by emergency messages, will mean decreasing the number of Reserved timeslots. This decrease will mean an increase of the Access timeslots. When using the idea proposed in paragraph 3.5.1, we will have a small delay for emergency messages.

3.5.4. Accessing emergency messages with higher power density

When allowing for emergency messages to access with a much higher power density than normal messages, we are making use of the capture effect on purpose (paragraph 3.2.2.1). Theoretically it is a simple solution, because one would guarantee a small delay. However, there will be a practical problem. The transceivers have to be adapted for transmitting with a very high power, which will be very expensive.
4. Interface description

4.1. The CAROLINE system architecture

As mentioned in chapter 2, to investigate many of the possible problems and features of the PREVU system, Philips is developing a testsite, called CAROLINE. The system architecture of this testsite can be subdivided into three parts: (i) the infra-structure (fig. 6.1), (ii) the in-car equipment (fig. 6.2), and (iii) the ETACS system that provides a full duplex radio frequency communication channel between (i) and (ii) (figures 4.1 and 4.2).

\[ 	ext{PSTN} \] = Public Switched Telephone Network

\[ * \] = including the Base Station part of the ETACS system

\textit{fig. 4.1 CAROLINE infra-structure including the BS part of the ETACS system.}
Interface description

RDS-TMC = Radio Data System - Traffic Message Channel
CARIN = CAR Information & Navigation system
AVL = Automatic Vehicle Location
* = including the Mobile Station part of the ETACS system

fig. 4.2 CAROLINE in-car equipment including the MS part of the ETACS system.

The infra-structure of the CAROLINE system.

Each Base Station System (BSS) shown in fig. 4.1 consists of two functional parts: the Base Station Controller (BSC, for handling the traffic center data flow), which is part of the CAROLINE infra-structure, and the Base Transmitter Station (BTS), which is a modified ETACS transceiver.

The CAROLINE infra-structure comprises three BSCs, a central traffic center (for handling the traffic statistics and situations), an emergency center (for example the police or a hospital) and a fleet-owner (for example a taxi-company). They are all interconnected via the public telephone network by means of modems.

The in-car equipment.

The Mobile Station System (MSS) shown in fig. 4.2 consists of two functional parts: the Mobile Station Controller (MSC), which is part of the CAROLINE in-car equipment, and the Mobile Transmitter Station (MTS), which is a modified ETACS transceiver.

The in-car equipment consists of a MSC that handles all kinds of applications such as RDS-
Interface description

TMC information, CARIN, floating car data, emergency call generator and AVL-location message generator. RDS-TMC provides (on a display) information about traffic jams, weather conditions, road conditions, etc.. CARIN is an in-vehicle autonomous navigation system. It will be made dynamic by means of the traffic flow data received from the traffic center. It will also provide the vehicle location to all the different applications and it is used to generate floating car data, which is information such as travelling times, etc., between different positions. A function for automatically generating emergency calls is included, which includes the vehicle position as part of its message. AVL-messages, extracted from CARIN, are intended for fleet-owners to see where their vehicles are.

The traffic information exchange within the infra-structure, and the data exchange in the in-car equipment, is important for the PREVU system as described in chapter 2. However, the traffic part of PREVU and the in-car equipment are beyond the scope of this report. More importantly, and with respect to the graduation subject, emphasis has been placed on the communication between the BSS and the MSS.

The communication between the BSS and the MSS.

The BSS and MSS are shown functionally in figures 4.3 a & b.

fig. 4.3 a & b BSS and MSS block diagram.
Interface description

Both the BSS and MSS consist of Transceivers and Controllers. The transceivers are full-duplex modified ETACS transceivers and the controllers are a Personal Computer (PC) and an interface board.

BTS and MTS.
In the transceiver the data is frequency modulated. There are 6 fixed duplex channels, of 30 kHz each around the 900 MHz frequency (subsection 2.2.2, the ETACS system). The transceiver has no "intelligence", i.e., the logical values on the input will become the same logical values on the output. The data sent by the transceiver has to be DC-free, because of its modulator and demodulator features. The difference between the transceivers of the BS and the MS is their carrier frequency, to perform the full duplex digital communication.

BSC and MSC.
The BSC has the same hardware as the MSC. However, the functionality of the BSC and MSC is different by means of different software.

The PC of the BSC handles the data flow to and from the central traffic center. In addition, it also handles the information flow to and from the different MSSs. Information from the MSs is received via a multiple access uplink channel. While information to the MSs is sent on a broadcast downlink channel. These information flows are realized through an interface board and a full-duplex transceiver. The PC of the MSC handles the data flow of the different mobile applications and the received data from the "closest" BS. The information flows handled by the PCs in both the uplink and the downlink, are performed by means of messages. These messages can be seen as "letters" within an envelope. The envelope, i.e., a routing header, tells the PC to which application(s) the letters have to be delivered.

The PC and the transceiver are connected by means of an interface board. The interface board hardware (with its limitations) already existed. The task was to realize the interface board software.

4.2. Functionality of the interface board

The main function of the interface board is to convert the asynchronous NRZ coded RS232 port of the PC to the synchronous DC-free lines of the transceiver. Another function it performs is the implementation of a time structure, which is needed in order to be able to implement the multiple access protocol. Finally, the multiple access protocol itself has also to be implemented in the interface board. In the interface board there is no "knowledge" about messages handled by the PC.

Performance of the DC-free spectrum.

To obtain the DC-free spectrum, FM0 is used (fig. 4.4). In FM0 encoding, also known as bi-phase space, a transition is present on every bit cell boundary and an additional transition may be present in the middle of the bit cell. In FM0 a "1" is sent as no transition in the center of the bit cell and a "0" is sent as a transition in the center of the bit cell, depending on the
Interface description

data. FM0 encoded data contains sufficient information to recover a clock from it. In other words: the clock is sent within the data and can be extracted from it at the receiver. With the received data and the clock, the original data can be recovered.

![Biphase space encoding (FM0).](image)

The drawback of using FM0 is that the net data rate can decrease by a factor 2, because a "0" bit is encoded by 2 level transitions.

The Serial Communication Controller (SCC) of the interface board (section 4.3) handles the PC-link as an asynchronous NRZ coded line and the SCC handles the transceiver-link as a synchronous FM0 coded line. The interface software moves the data from the PC-link SCC FIFO to the transceiver-link SCC FIFO, and vice versa. In this way the conversion from the asynchronous NRZ coded RS232 line to the synchronous FM0 coded line is performed, and vice versa.

**Specification of the timestructure.**

As described in chapter 3.3 the Reservation ALOHA protocol will be used. This protocol, which is an extension of the slotted ALOHA protocol, makes use of a timestructure. The timestructure consists of fixed timeslot lengths, separated by synchronization patterns (syncs, fig. 4.5). Because the syncs are overhead in the data stream, one wants to make the syncs as small as possible. The system timing has to be continuous, therefore the timestructure had to be implemented in the interface board.
Interface description

Fig. 4.5 Time-structure in the downlink.

There is another reason for implementing the time-structure. Due to interference and/or multipath fading, the receiver can lose synchronism. Therefore, some means of re-synchronizing is necessary. That is why it is advisable to add synchronization patterns in the transmissions.

CRC is not used in the downlink. The reason for this is explained by means of fig. 4.6.

Fig. 5.6 Message level on top on timeslot level.

The messages (layer 3 of the OSI model), which are one level higher than the timeslot structure level (layer 1+2 of the OSI model), have a variable length, marked with STRT (begin) and $TOP$ (end), and they have a CRC. The messages are handled by the PC, the timeslots are handled by the interface board. The message length is normally bigger than the timeslot length.

Suppose the timeslots have a CRC. This means one can detect bit errors, but one is not able to correct them. Since automatic repeat request by the MS can not be done because the BS broadcasts the information to all mobiles. If the interface board detects an error in a timeslot, it will not pass this timeslot to the PC. Suppose timeslot 4 in fig. 4.6 is destroyed and so will be thrown away. This will mean message 2 and message 3 will be destroyed. This diminish-
Interface description

es the throughput in the downlink. For the downlink it is important the MSs receive a big number of messages. Also, because the PC is more powerful than the interface board, the CRC is used in the messages instead of the timeslots.

**Specification of the multiple access protocol.**

The time structure exists only in the BS interface board, which ensures the continuously broadcast channel controls, amongst other things, the Reservation ALOHA protocol. Some bits are needed for this control (fig. 4.7). The MS interface board makes use of the time structure, made by the BS, to perform the Reservation ALOHA protocol. At the moment the MS receives a sync, it is allowed to send data (fig. 4.7).

\[ \tau_{\text{downlink}} \]

<table>
<thead>
<tr>
<th>sync</th>
<th>control</th>
<th>data</th>
<th>sync</th>
<th>control</th>
<th>data</th>
<th>etc.</th>
</tr>
</thead>
</table>

\[ \tau_{\text{uplink}} \]

| pre | data | post |

\[ \text{time} \]

\[ \text{guard time} \]

*fig. 4.7 Relation between downlink and uplink.*

The amount of data (including preamble and postamble) the MS sends after receiving a sync pattern, has to fit within one timeslot length to avoid interference between the timeslots (basic idea of the slotted ALOHA protocol, fig. 4.7). Making allowance for the switching on and switching off times of the transceiver, and from the different travelling times of the messages for the different MSs, a guard time before and after the data has to be used (fig. 4.7). Furthermore, the complete timeslot length $\tau_{\text{uplink}}$ has to be smaller than $\tau_{\text{downlink}}$ to avoid interference between the timeslots.

CRC needs to be used in the uplink, since one needs an acknowledge for implementing the Reservation ALOHA protocol. If no reservation is made (no acknowledge because of CRC error), the timeslot message in the uplink has to be resent.
Interface description

4.3. Interface equipment

4.3.1. Interface hardware

The block diagram of the interface board, which was provided by PHILIPS, is shown in figure 4.8.

![Block diagram of the interface board.](image)

The most important parts of the board are the Serial Communication Controller (SCC Z8530, [24]), the μ-processor (CPU Z80, [25]) and memory. The SCC consists of a transmitter and receiver to both the PC-link and the (FM0 coded) transceiver-link. Furthermore, a unique pattern (sync) is created and the CRC calculation is performed by the SCC. The remainder includes a part that converts the RS232 voltage levels to the SCC voltage levels, and dil-switches which are used as debugging tools.

**NOTE:** The interface board is not developed to handle interrupts. A circuit diagram of the interface board is shown in appendix A.

*SCC transmitter and receiver to both PC-link and transceiver-link.*

The SCC contains an asynchronous transmitter and receiver, each with a three character FIFO data buffer. Furthermore, the SCC contains a complete FM0 transmitter and receiver each with a three character FIFO data buffer.
Interface description

SCC creation of a unique pattern.
The SCC is able to make a unique 8 bit pattern (sync or flag). This is done by means of the Synchronous Data Link Control (SDLC) mode, in which frames of information are opened and closed by flags. The flag character has a bit pattern "01111110" and the sequence is unique because all data between the opening and the closing flags is prohibited from having more than 5 consecutive "1"s. The transmitter guarantees this by watching the transmit data stream and inserting a "0" after five consecutive ones, irrespective of character boundaries. In turn, the receiver searches the receive data stream for five consecutive "1"s and deletes the next bit if it is a "0". This technique is called zero-bit insertion (bit-stuffing).

SCC CRC calculation.
The 16 bit CCITT-CRC polynomial, \(x^{16} + x^{12} + x^5 + 1\), can be added in the SCC when using the SDLC mode. At the receiver side, the CRC between opening and closing flags will be calculated and compared with the received CRC. If they correspond, the CRC-error status bit will become false.

Micro-processor.
The 8 bit \(\mu\)-processor performs the task of transporting the characters from the PC side of the SCC to the TRX side of the SCC, and vice versa, via the memory. In addition, the implementation of the time structure and the Reservation ALOHA protocol is also done by the \(\mu\)-processor software.

Memory.
There are 8 kbytes of ROM, from address 0x0000 to 0xFFFF, containing the \(\mu\)-processor program with its constants. There are 8 kbytes of RAM, from address 0x8000 to 0x9FFF, containing the program’s variables and the buffers needed to perform the data exchange between PC and transceiver.

4.3.2. Interface software

The way that the software debugging was done is described by way of a description of the working environment. For clarity, the software implementation will be described in two parts. Firstly, a general overview will highlight the main features (section 4.3.2.2). This is then followed by a detailed software description (section 4.3.2.3), which will largely be of interest to software engineers and can be omitted by the (non-specialist) reader without loss of continuity.

4.3.2.1. Working environment

The NORAL SDT816 In-Circuit Emulator [26] was provided by PHILIPS for debugging. The development system environment is shown is fig. 4.9.
This configuration is called the terminal only mode (stand alone). Object code may be downloaded into the SDT816 overlay RAM or user target RAM from the PC.

The software for the Z80 μ-processor was written in the C-language. Because the Z80 μ-processor cannot be programmed in C directly, a cross-compiler for the conversion from C to Z80-assembler was needed. We used the Aztec C80/ROM cross-compiler [27]. Furthermore, because the SDT816 emulator makes use of the Intel hex format, conversion from Z80-assembler language to the Intel hex records also needed to be done. The conversion steps are shown in appendix B. The drawback of using a cross-compiler is that all the required conversion steps introduce a lot of software overhead.

For the Reservation ALOHA protocol, at the moment a MSC receives a flag from the BSC it is allowed to send. So there is a relationship between the BSC and MSC. That is why both the BSS and the MSS had to be used to test the written software. However, one does not need to use the transceivers (BTS and MTS) to test the written software. The transceivers can be seen as transparent boxes that only transport the logical values in both directions of the communication channel. This simplifies the test set-up, which is shown in figures 4.10 a & b.
There was only one SDT816 emulator (in-line software debugger) available. Therefore, suppose we were developing the MS program (fig. 4.10 a), then, for the MS we used the emulator to do the in-line testing and for the BS we used an interface board with the latest working version of the BS program in EPROM. To develop the BS program, we changed the interface boards (fig. 4.10 b). The interface board with the EPROM then contained the latest working version of the MS program.

4.3.2.2. Software implementation.

The software consists of two parts (fig. 4.11). The first part is the initialization of the SCC and the memory. The second part is an infinite polling loop in which the data transfer between PC-link and transceiver-link is performed. So, no interrupts are used. The functions to be performed in the transceiver link are the timestructure implementation in the downlink, and the Reservation ALOHA protocol implementation in both the uplink and the downlink.

In the first instance it was important to test the functionality of the protocol. For this reason, a decision was made to operate the system at a baudrate of 2400 instead of the full GSM-rate of 9600. In this way high baudrate problems were avoided.
Initialization

Initialization of the SCC.
The PC-link of the SCC is initialized to work in an asynchronous, eight bit per character, NRZ communication mode at 19200 baud. The transceiver-link of the SCC is initialized to work in a synchronous, FM0 coded, SDLC communication mode (section 4.3.1) at 2400 baud. It is part of the initialization that CRC is used or not. In the BS program no CRC was used, because of the implemented protocol on top of the timestructure. For the MS program CRC is used, because of the implemented Reservation ALOHA protocol (section 3.3).

Initialization of the memory.
To solve the timing problems in the program when transferring the data between the PC-link and the transceiver-link, a buffer is needed. Two FIFO buffers are initialized: one for the PC-link to transceiver-link direction (called toTRX buffer), and one for the transceiver-link to PC-link direction (called toPC buffer).

In addition, a dil-switch value is read which determines whether the interface program has to be the BS or the MS version. Finally, all the variables used for the implementation of the functions are initialized.

Infinite polling loop.
Interface description

By means of an infinite loop two full duplex channels are served: one duplex channel to the PC-link, one duplex channel to the transceiver link. A kind of RTS-CTS protocol is implemented to prevent buffer overflow.

The PC-link duplex channel.
If an eight bit character from the PC is available and one is allowed to receive data from the PC (controlled by means of the RTS-CTS type of protocol), it is put in the toTRX buffer. If data in the toPC buffer is available, it is given to the PC by means of eight bit characters.

A kind of RTS-CTS protocol is implemented to prevent buffer overflow. The complete RTS-CTS protocol is not implemented. The interface board can disable the PC to send data (RTS). The PC, however, cannot disable the interface board to send data (CTS). So, the CTS pin is not used. Suppose the PC is allowed to disable the interface board to send data to the PC (CTS is set), the toPC buffer will overflow after a certain period. The BS would be able to tell the MSs to stop sending when a buffer overflow is happening (via the broadcast channel), but a MS cannot ask the BS to stop sending, because all MSs are listening to the downlink broadcast channel.

Also, there is no need to use the CTS pin. The data rate between the PC and the interface board is higher (we chose \( \equiv 14000 \) baud) than the data rate between the interface board and the transceiver (we chose 2400 baud).

The transceiver-link duplex channel.
There are two versions for the transceiver link duplex channel. Firstly, the BS program, which sends the time structure and the control bits for the Reservation ALOHA protocol, and which receives data from the MSs. Secondly, the MS program, which sends data confirming the Reservation ALOHA protocol, and which receives data from the BS. The specifications of the data sent by both the BS and the MS will be given.

(1) Downlink to be performed by the BS.
The complete task the BS performs is shown in fig. 4.12.
Interface description

\[ \tau_{\text{downlink}} \]

<table>
<thead>
<tr>
<th>flag</th>
<th>control</th>
<th>data/dummy</th>
<th>overhead</th>
<th>flag</th>
<th>etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>control = 5 bits</td>
<td>data/dummy = 13*8 bits</td>
<td>overhead = 2 bits</td>
<td>flag = 8 bits</td>
<td>total = 119 bits</td>
</tr>
</tbody>
</table>

fig. 4.12 Downlink to be performed by the BS.

All bit functions, except the overhead bits, of fig. 4.12 have already been described in section 4.2. Out of the five control bits, just two are used: bit 0 to give the reservation status and bit 1 to give the status of the next block. This bit is needed, because the next block might be a dummy block, since even if there is no data to send, the timestructure still has to exist. The data/dummy length of the downlink is 13*8s bit because of the restriction that \( \tau_{\text{uplink}} \leq \tau_{\text{downlink}} \). The overhead is introduced because of a hardware imperfection of the SCC used. The flag at the end of the timeslot also acts as the opening flag of the next timeslot.

By means of figure 4.12 it is shown that the efficiency of the implemented downlink is 104/119, i.e., \( \equiv 87\% \). Taking care of zero-bit insertion, the value of 119 has to be increased by the number of zero-bit insertions, which decreases the efficiency.

(2) Uplink to be performed by the MS.
If the MS has data to send, the transceiver link is fed with a character at the moment a flag from the downlink is received. The complete task the MS performs is shown in fig. 4.13.
The boxes surrounded by thick lines in fig. 4.13 indicate time-periods and so are not real bits. The bit values shown in fig. 4.13 for these boxes are valid only in the case of a 2400 baudrate, i.e. one bit period is 1/2400 sec.

All bit functions of fig. 4.13 have already been described in section 4.2, except the "flutter" bit and the "O"-bit insertion time. The "flutter" time is introduced because the polling mode is used. It can happen that one main loop cycle has to be waited, before the transceiver will be switched on to transmit data. One loop cycle will mean about 1 bit period.

To make the flags unique it can happen that "0"-bits are inserted in the data and/or the CRC, which will make $\tau_{\text{uplink}}$ bigger. The zero-bit insertion time is needed to answer the demand that $\tau_{\text{uplink}} \leq \tau_{\text{downlink}}$. The value for this time was chosen to be 8 bits, because the probability of having more than 8 "0"-bit insertions is very small.

By means of figure 4.13 it is shown that the efficiency of the uplink timeslot is 64/119, i.e., $\approx 54\%$. 

---

**fig. 4.13 Uplink to be performed by the MS.**
4.3.2.3. Detailed description of the software implementation.

The flow chart of the BS and MS programs is shown in fig. 4.14.

* = functional difference between the MS and BS program

fig. 4.14 Flow chart of the BS and MS programs.
Interface description

Description of the functions.

Initialize PC-link.
Function name:  init_PC_link()

The right mode of communication is established by the bit values of the write registers (WR, [24]). Initialization of the SCC to the PC side of the interface board establishes a polled asynchronous communication mode between them. Some points of the initialization sequence WR values need to be stressed.

WR4.
Each character will have a start bit and two stop bits. Two stop bits have to be used, because the data rate between the PC and interface board is 19200 Baud. This is the highest rate the PC can handle. The result is that the successive characters will be "stuck" to each other and the asynchronous mode will look like the synchronous mode. To clarify the character boundaries, it is advisable to use the two stop bits. No parity bit is used, because having a "hard" wire connection means a very small bit error probability.

WR3 & WR5.
As a result of the 8 bit µ-processor, 8 bits per character will be used.

WR12.
WR12 contains the time constant for the baudrate generator, which delivers the clock frequency for both the transmitter and receiver, i.e., 19200 baud for the PC-link. The formula to calculate the time constant is given by:

\[
\text{Time Constant} = \frac{\text{Clock Frequency}}{2 \times (\text{Clock Mode}) \times (\text{Baud Rate})} - 2
\]

\[
= \frac{4915200}{2 \times (16) \times (19200)} - 2 = 6
\]

The clock frequency was chosen to be the system clock frequency. The net data rate between the PC and interface board, however, is only \((8/11) \times 19200 \equiv 14000\) baud. The factor \(8/11\) comes about, because each octet has one start bit and two stop bits overhead.

WR15.
In WR9 the master interrupt is disabled. This means it is not necessary to disable all possible interrupts of WR15 separately. However care should be taken! For example, if the Clear To Send (CTS) interrupt enable (IE) bit of WR15 is reset, the CTS bit in Read Register 0 (RR0) reports the current unlatched state of the CTS pin. Disabling of the master interrupt via WR9 does not mean that the CTS interrupt enable bit in WR15 is reset or set. Said differently: what state is reported by the CTS bit in RR0? To be sure that the unlatched state, and
not the latched state of the CTS pin is reported, one must reset the CTS IE bit in WR15. In WR15 all interrupts are disabled to solve the same kind of problems as mentioned in this example.

**Initialize dilswitch.**
Function name: init_dilswitch()

The dilswitch value at address 0xE005 is read and determines whether the interface program has to be the BS (value 1) or MS (value 2) version.

**Initialize transceiver-link.**
This functional part is marked with a * in fig. 4.14, which means there is a difference between the BS- and MS program.

**The BS program.**
Function name: init_TRX_link()

In initialize the transceiver-link, the correct mode of communication is established by the bit values of the write registers. Initialization of the SCC to the transceiver side of the interface board establishes a polled synchronous communication mode between them. Some points of the initialization sequence WR values need to be stressed.

WR4.
The Synchronous Data Link Control (SDLC) mode was chosen, because it performs a unique 8 bit flag without much overhead.

WR3 & WR5.
The address search mode of SDLC is not used, because the MSs identities must remain anonymous, while the BS makes no distinction between different MSs (it broadcasts its information).

CRC is not used in the downlink.

WR12.
The demand for the interface baudrate is at least 9600 baud. In first instance, it was important to test the functionality of the Reservation ALOHA protocol. We chose a 2400 baudrate.

When using the FM0 coding technique, the clock frequency needed for the SCC has to be 16 times the data rate. This clock is realized in hardware by dividing the system clock by 128. When using the SDLC mode, the clock mode is forced to one. With these values, the time constant needed for the baudrate generator, which delivers both the transmitter and receiver clock frequency, is:
Interface description

Time Constant = \frac{\text{Clock Frequency}}{2 \times (\text{Clock Mode}) \times (\text{Baud Rate})} - 2

= \frac{4915200}{128} - 2 = 6

WR15.
For the same reason as in the PC initialization link, WR15 has to be filled with all "0". If this is not done, the TXunderrun/EOM bit in RR0, for example, will not always be set.

The MS program.
Function name: init_TRX_link_CRC()

A major difference with the MS program is its use of CRC in the uplink (WR3, WR5, WR0). This is because, in the uplink one needs an acknowledge for implementing the Reservation ALOHA protocol. If no reservation is made (no acknowledge because of CRC error), the timeslot message in the uplink has to be resent.

Initialize buffers.
Function name: init_buffers()

To solve the timing problems in the program, buffers were needed. Two FIFO buffers, called the toPC and the toTRX buffer (fig. 4.15), were used.
Interface description

Data from the transceiver (TRX) is put in the toPC buffer by pointer putPC (most recently received data) and will be transmitted to the PC by means of pointer getPC (next data for transmission). The counterPC value gives the filled size of the toPC buffer. Data from the PC is put in the toTRX buffer by pointer putTRX (most recently received data) and will be transmitted to the transceiver by means of pointer getTRX (next data for transmission). The counterTRX value gives the filled size of the toTRX buffer.

For the toPC buffer, an extra pointer is needed only in the case of the BS program, called the help_putPC. The data from the MSs is sent with CRC. If no data is sent by the MSs, the receiver of the BS SCC will receive noise from the transceivers and will convert this to (wrong) data. At the BS receiver side, the data can be checked for the right CRC. If a CRC error has appeared, all data between opening and closing flags with the CRC error will be removed. This is done by means of pointer help_putPC and counter help_counterPC (fig. 4.15).

Pointer help_putPC is the pointer to the most recently received error-free data. Pointer putPC is the pointer to the most recently received incoming data. If a CRC error has appeared, the data between help_putPC and putPC will be deleted.
Interface description

*Initialize global variables.*
Function name: `init_flags()`

In this function the global variables (like the downlink control bits, CRC status bits, etc.), which are needed for implementing the timestructure and the Reservation ALOHA protocol, are initialized. Global variables instead of functions that return values were used, because it made debugging easier (see software working environment).

*Check buffer size.*
Function name: `check_toTRX_size()`
Function calls: `rts_assert()`
`rts_deassert`

If the toTRX buffer (fig. 4.15) is less than 20% full, the Request To Send (RTS) pin is set, by means of function `rts_assert()`, to ask the PC to send data. If the toTRX buffer is more than 80% full, the RTS pin is reset by means of function `rts_deassert`, to disable the PC from sending data.

*Receive data from PC.*
Function name: `receive_PC()`

If data from the PC is available, it is put in the toTRX buffer (fig. 4.15).

*Transmit data to transceiver.*

*The BS program.*
Function name: `transmit_TRX_base()`

The transceiver link is fed with a character. Depending on the state in which the BS is while transmitting, it performs different subtasks. The complete task the BS performs is shown in fig. 4.12.

The control character consists of five bits, however, only two of them are used. Bit 0 is the reservation bit that gives the acknowledge to the MSs. An acknowledge means the reservation bit is set (section 3.3). Bit 1 is the bit that gives the state of the next block; data or dummy. The control character consists of five bits, since it is only possible to use less than five bits per character if these are sent at the end of a timeslot. The number of bits will then be given by means of the residue code. Using a control character at the end of a timeslot makes programming more difficult, so the control character was placed at the beginning of the timeslot.

If the PC has given enough data to the interface board, the data/dummy block will be filled with data. If the data from the PC is less then the data/dummy block size, the data/dummy block will be filled with dummy data. This dummy data is needed to realize the timestructure,
Interface description

even if there is no data to send. The drawback of this solution is that if message N is not an integer number of timeslots, the last part of this message (called the "remainder", which is smaller than one timeslot length) will not be sent until message N+1 is sent. If message N+1 plus the remainder of message N is not an integer number of timeslots, the remainder of message N+1 will not be sent until message N+2 is sent, and so on.

The size of the data/dummy block depends on the size of one uplink timeslot, which must fit within the size of one downlink timeslot ($\tau_{\text{uplink}} \leq \tau_{\text{downlink}}$). The size of one uplink timeslot is 119 bits (fig. 4.13). So, the size of one downlink timeslot has to be at least 119 bits. Because the flag length, control length and overhead length of the downlink timeslots are fixed, the data/dummy length is chosen to be 13*8 bit (fig. 4.12). By choosing this value, the complete downlink timeslot length will be 119 bits in the case of no zero-bit insertion. With zero-bit insertion, the downlink timeslot length will be bigger, but this does not contravene the demand that $\tau_{\text{uplink}} \leq \tau_{\text{downlink}}$.

The data/dummy block consists of 8 bit characters. Note that the change from 5 bits per character to 8 bits per character has to be done just before the first 8 bit character has been sent.

The last two bits of the CRC are never transferred to the receive data FIFO of the SCC. Because we do not use the CRC in the downlink, the last two bits of the data/dummy block will not be recoverable. By means of sending two extra dummy bits at the end of the timeslot, the data will not be destroyed. However, these two bits are overhead in the downlink.

The flag "01111110" at the end of the timeslot is also the opening flag of the next timeslot. It is realized in software by means of sending no data during the time the eight bit flag is being sent. Disabling of sending data will mean a delay. However, in the meanwhile, the other tasks, like receiving data, still have to be performed. Because we are not making use of CRC in the downlink, the TXunderrun/EOM bit of RRO will not be set. So this bit can not be used to determine that exactly one flag has been sent.

The delay is realized by means of a counter, that decreases every new loop cycle of the main program. Because the execution of every loop cycle is not fixed, the delay will also not be fixed. The initial value of the counter has to be just small enough, that the SCC starts to send a flag. When the SCC has started to send a flag, it will always send the complete flag. By means of this method, it will be possible to exactly send one flag, even while the delay is not fixed. Note that if the program is developed further (size may change), the initial counter value will have to be changed appropriately.
Interface description

The MS program.

Function name: transmit_TRX_mob()

If the MS has data to send, the transceiver link is fed with a character at the moment a flag is received. Depending on the state in which the MS is while transmitting, it performs different subtasks. The complete task the MS performs is shown in fig. 4.13.

The "flutter" time is introduced because the polling mode is used. It can happen that one main loop cycle has to be waited, before the transceiver will be switched on to transmit data. One loop cycle takes about 600 \( \mu \text{sec} \) (section 5.2), i.e., 1 bit duration with a 2400 baudrate.

The transceiver is switched-on and -off via the Data Terminal Ready (DTR) pin of the interface board. The time between receiving a signal at the BS after switching on the transceiver of the MS and the time between receiving hardly any signal (namely the transceiver is never switched off 100\% when using the DTR pin) at the BS after switching off the transceiver of the MS are shown in fig. 4.16. Note that these are back-to-back measurements, and are independent of MS-BS propagation time delay.

![Transceiver switching-on and -off times](image)

The complete switching on delay is about 450 \( \mu \text{sec} \), which is roughly one bit duration. After this period, the data can be transmitted. However, in practice it appeared that the first data sent was not received error free. At least 7 extra "dummy" bits (9 bits were chosen to get a better performance) were needed. There may be two reasons for this. Firstly, and probably most importantly, the receiver has to be bit synchronized before it can receive data. Secondly, the transceivers have to be initialized completely, including its \( \mu \)-processor. Choosing nine bits also includes the guard time needed because of different propagation times \( t_{\text{path}} \) (section 2.2.2, ETACS). The "dummy" bits are chosen to be all "1s".

The first flag is the preamble of the timeslot. Note that the receiver side recognizes a difference between an opening flag and a closing flag.
The data is sent by means of 8 bit characters, because of the 8 bit µ-processor. GSM uses about 57 information bits in one timeslot. For reasons of compatibility with GSM, the number of databits was chosen to be at most 8*8 bit. Now if it happens, that after sending less than 8 characters, there is no more data to send, then the data burst will be completed (i.e., end flag etc. appended and transmitted).

The 16 bit CCITT-CRC polynomial, \( x^{16} + x^{12} + x^5 + 1 \), was chosen to perform the error detection. This CRC is performed in hardware by the SCC.

The zero bit insertion time was needed to make the flags unique. After 5 consecutive "1s" a "0"-bit will be inserted, which increases the number of bits sent. The "0"-bits can be inserted in both the data bits and the CRC bits. In the worst case situation \((64 + 16)/5 = 16 \) "0"-bits can be inserted. Assuming completely random data, the probability of this worst case situation arising is \((1/2)^{80} \approx 8*10^{-25}\). The average number of inserted bits, however, is much smaller. In appendix C the probability of having N inserted bits in one timeslot is calculated. A "0"-bit insertion time of 8 bits was deemed to be sufficient, because the probability of having more than 8 "0"-bit insertions is very small (appendix C).

The switch TRX-off time of 4 bits consists of three different parts: a TRX-wait time, a TRX-delay time and a TRX guard time.

TRX-wait time.
As soon as the MS transceiver is switched off, no bits can be detected at the receiver side of the BS, even though a high signal strength may still be received (fig. 4.16). This occurs, because within the transmitter there is a one bit period delay, therefore, one has to wait one bit period before switching off the transceiver, to be sure that the last bit is really sent.

TRX-delay time.
As shown in fig. 4.16, the measured switching-off delay of the transceiver was about 700 µs, i.e., 2 bit periods. During this time no information bits are sent, but the transmitted signal can still interfere with other MSs.

TRX-guard time.
An extra guard time of one bit period is used; half a bit period for the TRX wait time, half a period for the switch TRX-off time. This was done to be sure that these tasks are performed completely, thereby improving the performance.

Receive data from transceiver.

The BS program.
Function name: receive_TRX_base()
Function calls: remove_CRC()

The data from the transceiver link is put in the toPC buffer of the BS if its CRC is true. If the
Interface description

CRC is false, the complete timeslot will be deleted. If no MS is sending data, the BS receiver is in hunt mode, i.e., searching for flags. As soon as a flag is received, the SCC is out of hunt mode and the BS program is able to delete all received "noise data" before the flag (see function initialize_buffers).

By means of the end of frame bit in RR1 [24], the CRC position can be found. One can not make use of the fact that the CRC of the received "noise data", before the first flag, is false, because the end of frame bit will not be set after receiving an opening flag. The end of frame bit will only be set after receiving an ending flag.

To prevent the toPC buffer overflowing with "noise data", one must always delete the last received data once its length has become bigger than the maximum MS timeslot information length, i.e., the uplink data length plus the uplink CRC length (fig. 4.13).

Because the baudrate to the transceiver link is less than the baudrate to the PC link, the toPC buffer will not overflow (see function check_buffer_size).

The received flags are not transferred as data to the receive data FIFO of the SCC. The received CRC, however, is transferred as data to the receive data FIFO of the SCC. The μ-processor has to remove these 16 CRC bits in the toPC buffer. This is done by means of function remove_CRC().

The MS program.
Function name: receive_TRX_mob()
Function call: check_frame_end()

The data from the transceiver link is put in the toPC buffer of the MS. Function check_frame_end() returns that a flag has been received. This is needed for two reasons. Firstly, to tell the receive_TRX_mob() function that the next received character has a 5 bit length instead of 8 bit. Note that changing the character length has to be done just before the character is received. Secondly, to tell the transmit_TRX_mob() function that it is allowed to send data.

The control character is investigated and some status bits are set. Up to now, only the data/dummy control bit is used. If this bit returns a "0", all data up to the closing flag will not be passed to the toPC buffer. If the bit returns a "1", all data up to the closing flag will always be put in the toPC buffer. Because CRC is not used at the timeslot level, the PC, working at the message level, has to decide to delete the message if there is a CRC error.

The reservation bit will be used to give the acknowledge, so the transmit_TRX_mob() function knows in which timeslot it is allowed to send.

Transmit data to PC.
Function name: transmit_PC()
Interface description

The PC link is fed with a character, if data in the toPC buffer is available.

*Print error.*

Function name: print_error()

This function is used for debugging. It prints the "error" warning to the PC if the dilswitch has an indefinite value.
5. Results

5.1 Results of the implemented protocol

In theory, the Reservation ALOHA protocol seems to be a good multiple access protocol that satisfies the CAROLINE specifications. The full Reservation ALOHA protocol, however, has not yet been implemented. What has been implemented up till now is shown in fig. 4.12 for the downlink and fig. 4.13 for the uplink. It is a slotted ALOHA protocol without acknowledgment. The baudrate is 2400 bit/s.

Discussion of the results.

The downlink.

It is shown in fig. 4.12, that some evident drawbacks considerably diminish the downlink efficiency. There are five bits to control the protocol and two overhead bits due to the imperfection of the SCC. Furthermore, there is an eight bit flag. These 15 overhead bits decreased the net baud rate by a factor 104/129 ≈ 0.87. Taking care of zero-bit insertion, the value of 119 has to be increased by the number of zero-bit insertions, which decreases the efficiency.

As a result of using a unique, 8 bit flag for indicating the time at which the MS should transmit, zero bit insertion had to be done. Zero bit insertion makes the timeslot length variable. The variation in timeslot length, however, is rather small, assuming random data (appendix C). This variation in the downlink timeslot length, however, does not influence the operation of the slotted ALOHA protocol. The MSs, immediately after receiving a flag, are allowed to send. If, because of zero bit insertion, the flag will be "delayed", all MSs will receive this "delayed" flag, and so all MSs will still be allowed to send at the same time.

The uplink overhead.

In the uplink, there is a big drawback, because of overhead (fig. 4.13). The overhead can be divided into three different parts: overhead start-time, communication overhead and overhead end-time. Using a 2400 baud rate this complete overhead was 55 bit. So, the efficiency of the implemented uplink timeslot was 64/119, i.e., ≈ 54%.

Overhead start-time.

The overhead time at the beginning of a timeslot consist of a "flutter" time, a guard time, a switch-on time of the transceiver and a wait time for the transceiver before sending the first flag. Assuming a 2400 baud rate, the sum of these times will be 11 bit periods.

One can decrease the "flutter" time by means of interrupts, since this means that one does not have to wait for one main loop cycle. Using a powerful μ-processor, which serves the interrupts very quickly (i.e., there is hardly an interrupt queue), will decrease the "flutter" time considerably.
Results

A guard time was needed to solve the difference in propagation time delays (section 2.2). Within GSM a guard time period already exists, and so, ultimately, this will not be needed in the protocol.

The long switching times can be reduced by using other transceivers which are specially made to avoid this problem. The transceivers used were developed for speech, and were not intended to switch on and off very quickly.

The SCC transmitter had to wait before sending the first flag, probably because the SCC receiver needed several bits before it was bit-synchronized. These first bits were not received correctly. Using special customized transceivers will solve this problem.

Communication overhead.
The communication overhead consists of two flags, the CRC and the "0"-bit insertion time, which in total was 40 bits. The two flags are always needed to give the beginning and end of the timeslot. To make a unique flag, it is very hard to use less than 8 bits without needing a lot of overhead to code the data.

Instead of the 16 bit CRC for example, a parity bit for each eight bit character could be used. Then only 8 bits would be needed. This, however, is a very weak error detection technique.

To ensure no interference between the timeslots, "0"-bit insertion was needed. The value of eight bits can be minimized, but this would increase the probability of interference (appendix C). The "0"-bit insertion overhead is communication overhead, but it is expressed in time (fig. 4.13).

Overhead end-time.
The overhead time at the end of a timeslot consists of a wait time before switching the transceiver off, a switching-off time of the transceiver and a guard time. For a 2400 baudrate, the sum of these times was 4 bit durations. The wait and switching-off times of the transceiver are similar to the switching-on times discussed above and can be treated in a similar manner.

The uplink random generator.
The random generator needed to realize the slotted ALOHA protocol was implemented in the PC. The PC randomly gives messages to the interface board. The MS interface board, as soon as it receives data, sends its data immediately after receiving a flag, so this is done randomly too. The BS interface board checks the received CRC and sends an acknowledge (reservation) back to the MS if the CRC is true. However, up to now the MS does not resend the timeslot if no acknowledge is received.

The uplink throughput.
In section 3.4, the net throughput, which is realistic for the CAROLINE testsite was calculated:
Results

\[
\text{net throughput}_{\text{timestruc}} = \frac{\delta}{\delta + \epsilon} \cdot \frac{\beta + 1}{\beta + 1/\alpha}
\]

where

- \(\alpha\) = probability of success,
- \(\beta + 1\) = average number of reserved timeslots per message,
- \(\delta\) = timeslot length in bits,
- \(\epsilon\) = sync pattern length in bits.

This formula can be used to calculate the net throughput for the CAROLINE testsite, taking into account the complete overhead. The variable \(\epsilon\) however is not only the sync pattern length, but the complete overhead needed in one uplink timeslot. The results of these calculations are shown in table 5.1.

Table 5.1 Net throughput_{CAROLINE} calculations depending on message length. 
\((\alpha = 0.37, \delta = 64, \epsilon = 55)\)

<table>
<thead>
<tr>
<th>message length in timeslots ((\beta + 1))</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentage throughput</td>
<td>20</td>
<td>29</td>
<td>34</td>
<td>38</td>
<td>40</td>
<td>46</td>
<td>50</td>
</tr>
</tbody>
</table>

It is clear that it is impossible to reach a net throughput_{CAROLINE} above \(\delta/(\delta + \epsilon) \equiv 54\%\), i.e., 1200 baud. This seems to be a really low value, however, note that the theoretical values calculated in section 3.4 did not take into account the communication overhead.

A clearer comparison between the throughput values for the CAROLINE multiple access protocol and the theoretical throughput values, i.e., ignoring the communication overhead, but including the overhead time, can be done by means of changing the value \(\epsilon\) from 55 to 15 bits. The results of these calculations are shown in table 5.2.

Table 5.2 "Gross" throughput_{CAROLINE} calculations depending on message length. 
\((\alpha = 0.37, \delta = 64, \epsilon = 15)\)

<table>
<thead>
<tr>
<th>message length in timeslots ((\beta + 1))</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentage throughput</td>
<td>30</td>
<td>44</td>
<td>50</td>
<td>57</td>
<td>60</td>
<td>69</td>
<td>80</td>
</tr>
</tbody>
</table>

It is clear that it is impossible to reach a "gross" throughput_{CAROLINE} above \(\delta/(\delta + \epsilon) \equiv 81\%\).
Results

5.2. Anticipated performance with an increased baudrate

To show the features when using the GSM system, the net baudrate, i.e., the baudrate without the communication overhead, has to be 9600 baud. This means that the "gross" baudrate has to become on the order of 19200 baud. Increasing the baudrate will mean more practical problems, i.e., the slow switching transceivers, need for a DC-free spectrum, buffer overflow and a lack of processing power.

Slow switching transceivers.
The throughput values of tables 5.1 and 5.2 were calculated when using a 2400 baudrate. If the baudrate is increased to 19200, i.e., a factor of 8, the absolute overhead time will not change, but the overhead time, expressed in bitperiods, will also be increased by a factor of 8 to 8*15=120. Using 64 databits with already 120 overhead bits, mainly as a result of slow switching transceivers, will not be acceptable. The net throughput will (in this case) never be bigger than \(\frac{\delta}{(\delta + \epsilon)} \approx 35\%\), i.e., 6720 baud.

Requirements for a DC-free spectrum.
The signalling rate in one ETACS channel is required to be 19200 baud. The spectrum given to the transceivers has to be DC-free. Demanding a DC-free spectrum and having available a 19200 baud signaling rate, will demand a significantly larger bandwidth. One always needs some overhead to realize the DC-free spectrum. The coding technique we are using, called FM0 (fig. 4.4) decreases the net baudrate in the worst case situation a factor of 2. Because of this, the highest possible gross baudrate will be 9600.

Buffer overflow.
The net baudrate between the interface board and the PC is \((8/11)*19200 \equiv 14000\) baud (section 4.3.2.3). Suppose the net baudrate between transceiver and interface board becomes higher than 14000 baud. This will generate a problem. The toPC buffer will overflow, because there is no RTS-CTS (or corresponding) protocol between transceiver and interface board (section 4.3.2.2).

Lack of Z80 processing power.
The Z80 \(\mu\)-processor serves four channels, two simplex (transmit and receive) channels to the PC-link and two (transmit and receive) channels to the transceiver link. In the worst case situation, all four channels have to be served. Having a 19200 baudrate program for both the PC-link and the transceiver-link (so each channel has the same baudrate program), one can calculate the maximum time, \(T\), in which one main loop cycle has to be performed, if there is no buffering of the channels.

By baudrate program is meant the data rate handled by the program. The flags and the start and stop bits are not loaded in the three character FIFO of the SCC, so the program does not handle these data. Using a 19200 "gross" baudrate for the PC-link means its baudrate program is \((8/11)*19200 \equiv 14000\) baud, because of the one start and two stop bits. Using a 2400...
"gross" baudrate for the transceiver-link means its maximum net downlink baudrate program is \((111/119)2400 \equiv 2250\) baud because of the 8 bit flag. During the overhead times shown in figure 4.13, it can be possible that the program has to handle data (for example noise data) and so the maximum net uplink baudrate program is \((103/119)2400 \equiv 2000\) baud because of the two 8 bit flags.

If one loop takes more than \(T\) seconds, it will be possible for a 19200 baudrate program to serve all four channels. Within \(T\) seconds, in the worst case situation, all four channels have to be served. Suppose each channel handles 8 bit characters, i.e., 32 bits have to be served within one main loop by the program. This means that one bit has to be served within \(T/32\) seconds. If one bit takes \(T/32\) seconds, the bitrate is \(32/T\) baud. This bitrate has to be higher than \(4 \times 19200\) baud to serve each 19200 baud channel, i.e., \(32/T \geq 4 \times 19200\)

\[
T \leq 32/(4 \times 19200) \equiv 416 \mu\text{sec}.
\]

In this application the value of \(T\) has to be even smaller. In the worst case situation, within one main loop cycle only 23 bits have to be served: two 8 bit channels to the PC-link, one 5 bit channel while receiving a 5 bit control character, and one 2 bit channel while transmitting the two bit overhead. Now, it is rather difficult to calculate the value of \(T\) exactly.

The value of \(T\) is determined by the smallest served character. In this case one should expect that this is the 2 bits overhead character (fig. 4.12). But a problem arises only at the transmit channel and not at the receive channel, because the program does not handle the last two received bits of the timeslot (section 4.3.2.2), so these 2 bits are not part of the baudrate program. This problem at the transmit channel is not large. If the transmit channel is not served in time after transmission of the last two overhead bits, two flags instead of one will be sent. The Reservation ALOHA protocol can still be implemented, with the result that only the downlink overhead will be increased.

In this application, the value of \(T\) is determined by the control character of 5 bits (fig. 4.12). Because this character is almost half as short as the 8 bit character, the demand that \(T \leq 416 \mu\text{sec}\) has to be changed to \(T \leq (416/2) \mu\text{sec} = 208 \mu\text{sec}.

In the average situation, in the till the most recently developed program, one main loop cycle takes about 3000 clock cycles. Using a 5 MHz system clock, this means that one average main loop cycle takes \(3000/(5 \times 10^6) = 600 \mu\text{sec}\). Analysis of the program showed that a worst case situation may well take twice this time, i.e., \(1200 \mu\text{sec}\). This value is 6 times higher than the demand to serve the channel at a 19200 baudrate program. So, because of a lack of processing power in the Z80 µ-processor, the maximum baudrate program will be \(19200/6 = 3200\) baud for each channel. If one channel has a higher baudrate program, one main loop cycle is determined by this channel because its characters have to be served in time.

Using the same "gross" baudrate in both the PC-link and the transceiver-link can generate a
Results

buffer overflow, if the baudrate\text{program} in the transceiver-link is higher than the net baudrate\text{program} in the PC-link (section 4.3.2.2). This is why the PC-link "gross" baudrate is chosen to be higher than the transceiver-link "gross" baudrate. In practice, it appeared that the \mu\text{-}processor can still handle a 19200 "gross" baudrate for the PC-link (i.e. \(\approx 14000\) baudrate\text{program}) and a 2400 "gross" baudrate for the transceiver-link (i.e., \(\approx 2000\) baudrate\text{program} for the uplink and \(\approx 2250\) baudrate\text{program} for the downlink).

This seems to be in contrast with the fact that one loop cycle is determined by the highest baudrate\text{program} channel. As explained before, a 14000 baudrate\text{program} channel would not be possible. However, there is buffering, i.e., a 3 character FIFO, in the SCC. The possibility of obtaining a worst case main loop cycle time on three consecutive occasions is exceedingly small. With the 3 character FIFO one therefore has to allow for the average main loop cycle time and not the worst case.

5.3 Anticipated performance with GSM

When using GSM, the overheads will be much less. The overhead time (fig. 4.13) will not exist as the transceiver switching times are much smaller and are taken care of within GSM by means of guards at the beginning and the end of each timeslot. The communication overhead, in both the downlink and the uplink, to perform the multiple access protocol is delivered by GSM already, so it will not be needed again while implementing the Reservation ALOHA protocol.
6. Recommendations

In the CAROLINE testsite, one wants to test the features when using the GSM system. The baudrate of GSM is 9600. So, the demand for the "gross" baudrate within the CAROLINE testsite is on the order of 19200 baud. With the currently available tools, it will not be possible to reach this baudrate for three reasons: the used coding technique to perform the DC-free spectrum, lack of Z80 processing power and the introduced overhead to implement the multiple access protocol. The recommendations to improve these imperfections as much as possible are discussed in this chapter.

The complete multiple access protocol is not yet implemented. The complete description of the functions to be performed for completion of the Reservation ALOHA protocol are also shown in this chapter.

6.1 Increasing the baudrate

*Improvement of the coding technique.*

The spectrum to be given to the ETACS transceivers has to be DC-free. Therefore, this excludes NRZ data. The overhead needed to perform a DC-free spectrum depends on the chosen coding technique. The coding technique we used, biphase mark (FM0), results in a narrower bandwidth than would be obtained with Manchester encoding for example. However, it is advisable to investigate whether further improvements can be gained, although it is expected that large improvements will not be possible from coding alone. Multi-level modulation schemes [28], can provide the improvements but would require significant changes of the transceivers. For the time being, this would only be considered as a final resort.

*Improvement of the lack of the Z80 processing power.*

There is a problem in reaching a 19200 baudrate due to the fact that the Z80 is not a very fast \(\mu\)-processor. However, there are some solutions to overcome this problem by optimizing the usage of the Z80 \(\mu\)-processor:

- Using the same baudrate for every channel.
  As explained in section 5.2, the time to complete the main loop is governed by the highest baudrate. If one uses the same baudrate in all the channels, then the processor time is distributed evenly over the different channels. Hence the overall performance of the interface increases.

- Using a higher clock-rate.
  There are some Z80 \(\mu\)-processors that can work with a 10 MHz clock frequency instead of the 5 MHz clock frequency used in this study. This can immediately increase the baudrate for each channel by a factor two.
**Recommendations**

- **Optimize the software.**

A lot of unnecessary software is generated due to the cross compilation process. Figure 6.1 shows the performance analysis of both the MS and BS programs. The spaces between the addresses are equal.

![Address Analysis Diagram](image)

- **= Mobile Station program.**
- **= Base Station program.**

**fig. 6.1 Performance analysis of the program.**

Figure 6.1 shows that the 'critical' part of the program is between addresses 0xA51 and 0x0B00. The code between these addresses consists of library functions called by the cross-compiler. Examples of these library functions are compare operations and save register operations after a function call.

If one wrote the program in Z80 assembler, most of these library function calls would be unnecessary and the remainder of these library functions can be optimized. It is expected that in this way the program code could be reduced by more than a factor of two. So, the execution of one main loop cycle can be increased by a factor of two, which also increases the used
Recommendations

baudrate for each channel by a factor of two.

- **Always using an eight bit character for each channel.**
  
  As explained before, serving eight bit characters in the main loop cycle will take as much time as serving, for example, two bit characters. In this example the served channel baudrate, however, is four times smaller. Always using eight bit characters will optimize the usage of an eight bit μ-processor.

  The drawback of this solution is the overhead of transmitting eight bits while only two of them are really used. One has to try to take together all characters smaller than eight bits to create eight bit characters. In the implemented worst case situation, instead of serving 23 bits in one loop (two times 8 bits for the PC-link, 2 bit for the BS overhead and 5 bits for the MS control character), 32 bits (four times 8 bits in both the PC-link and the transceiver-link) are served in one loop. It is expected that the application of this solution can increase the baudrate for each channel a factor 32/23 ≅ 1.4.

- **Using two interface boards in parallel.**
  
  Because the transmitter and receiver functions are almost completely independent, splitting up the interface board functions into two parts can be easily performed. It is possible to use one interface board to perform the transmitter functions, and another to perform the receiver functions.

  The only relation between the transmitter and receiver functional parts of the interface board, is that a character is allowed to be transmitted after a sync has been received. This can easily be performed by connecting the sync output pin of the receiver to the sync input pin of the transmitter.

  Instead of serving four channels, one interface board now just has to serve two channels. So the baudrate in one channel can be increased by a factor of two by means of this solution.

- **Using a RTS-CTS (or look alike) protocol more often.**
  
  By means of using a kind of RTS-CTS protocol, it is possible to disable one channel during a certain period. Decreasing the number of channels that have to be served, decreases one main loop cycle time and so the channel baudrate can be increased. However, note that this method has to be done on the correct times, otherwise it will influence the system performance. Therefore, it is advisable to use this method as little as possible.

  By means of using all the previously described methods, it may be possible to increase the baudrate of each channel by a factor of 8. Instead of using 2400 baud for the transceiver link, 19200 baud could be used. However, the Reservation ALOHA protocol is not completely implemented yet. The Z80 μ-processor will have to perform more tasks. Because one is already working at the processing power limit, implementation of more functionality to be performed by the μ-processor will result in a shortage of the processing power. That is why it is recommended to search for a more powerful μ-processor to implement the Reservation ALOHA protocol and its required timestructure.
Recommendations

8.2. Decreasing the overhead while implementing the protocol

A lot of the overhead is introduced because of the slow switching transceivers and the use of the SDLC protocol. This overhead decreases the throughput.

Decreasing the overhead caused by the slow switching transceivers can be performed by means of faster switching transceivers. The GSM transceivers will meet these switching requirements. It is recommended to wait for these GSM transceivers. Up to now, the overhead due to the slow switching transceivers can be decreased by just one bit. The switch on delay of the transceiver is the same for all MSs and so during this time there is no interference between the MSs. One can make use of this fact, so the timeslot length can become one bit larger.

The SDLC protocol makes use of zero-bit insertion. The zero-bit insertion technique makes the timeslot length in both the uplink and the downlink variable, because the data length between the flags, excluding zero-bit insertion, is fixed. If the data length between the flags, including zero-bit insertion is fixed, the throughput will be increased for two reasons. Firstly, the downlink timeslot length has to be fixed to get an optimal throughput [3]. Secondly, if within a timeslot few zero-bit insertions are required, the reserved zero-bit insertion overhead (fig. 4.13) can be filled with data. The strategy to accomplish this is described in the following.

The data and CRC length plus the number of zero-bit insertions is equal to $\lambda$. In the case of 64 databits and 16 CRC bits, the maximum number of zero bit insertions is 16 (appendix C), so $\lambda_{wc}$ (worst case) is equal to 96 bits. This figure ($\lambda_{wc}$) is then used as the fixed length between the flags. Usually there will not be 16 zero bit insertions and the unused zero bit are used to carry extra data, so reducing the overhead. The processor is used to calculate the number of insertions and hence the number of extra data bits to be transferred.

First $\lambda$ is calculated with number of data bits equal to 64. If $\lambda$ is less than 96 bits, then $\lambda$ equal to 65 data bits will be calculated. If this $\lambda$ is still less than 96 bits, then the process is repeated until $\lambda$ with the highest number of data bits and with a length that is less than or equal to 96 bits is found. The timeslot will be send with this found $\lambda$. The uplink timeslot length is still not fixed, because $\lambda$ is not always equal to 96, but the reserved zero-bit insertion overhead (fig. 4.13) is more efficiently used by data, when using this method.

8.3. Continuous sending of the complete downlink message

If the data from the PC is less then the data/dummy block size, the data/dummy block will be filled with dummy data. This dummy data is needed to maintain the timesstructure, even if there is no data to send. The drawback of this solution is that if message N is not an integer number of timeslots, the last part of this message (called the "remainder", which is smaller than one timeslot length) will not be sent until message N+1 is sent. If message N+1 plus the remainder of message N is not an integer number of timeslots, the remainder of message
Recommendations

N+1 will not be sent until message N+2 is sent, etc.. It is required to change this solution: sending data in the downlink as soon as there is data and filling up the timeslot with dummy data in case there is not enough data.

8.4. Completion of the Reservation ALOHA protocol

What has been implemented up to now is an uncontrolled slotted ALOHA protocol without acknowledge. There are some steps to be performed for completion of the Reservation ALOHA protocol. Firstly, the BS acknowledgment has to be used. Secondly, the protocol has to be controlled in order to work at its optimum. Thirdly, the reservation process has to be performed. Finally, the solution for emergency messages has to be implemented.

Acknowledgment used in the protocol.

The BS interface board sends an acknowledge back if it received a true CRC. However, up to now the MS interface board does not re-send the timeslot if no acknowledge has been received. The access timeslot with the CRC error should be re-sent after a certain random period. Because the random generator is performed by the PC, the interface board has to ask the PC to re-send the timeslot. However, the PC has no knowledge about timeslots, it only has knowledge about messages. The interface board, on the other hand has knowledge about timeslots but no knowledge about messages. The protocol to perform the connection between the PC and interface board, in respect to the random generator is rather complex. Therefore, it will be much easier to perform the random generator in the interface board.

After implementation of the random generator within the interface, an uncontrolled slotted ALOHA protocol should be implemented. The way in which the acknowledge should be performed for the Reservation ALOHA protocol is explained in the description of the reservation process.

Control of the protocol.

For controlling the protocol, the fraction of timeslots that were empty during the recent past, has to be known by the BS (section 3.2.2.1). This fraction can be estimated by means of the squelch of the transceiver. If the offered load changes, the BS has to re-send extra control bits, to tell all MSs the new mean delay before (re)transmission.

Reservation process.

As soon as a user has a message to transmit, it listens to the downlink for an indication that the next timeslot (e.g. number n) is free for uplink transmission. Within this timeslot all users with a message transmit the first part (corresponding to the length of a timeslot) of the message with a probability P. P is broadcasted by the BS. If only one user transmits in timeslot n, he will receive an acknowledge in timeslot n+1 if there was no collision and this means that timeslot n+2 will be reserved if this user indicated in timeslot n that he wants to transmit more data. This user will thus know from this reservation that his data in timeslot n...
Recommendations

has been received properly. If no one transmits or two or more users transmit and interfere, then timeslot n+2 will be free for anyone to transmit.

So a given user with a message of several timeslots transmit in slots n, n+2, n+4, etc. The same strategy can be used for the sequence of slots n+1, n+3, n+5, etc. As explained, an interleaving of two sequences is necessary to give the central transceiver and the users time to do the necessary processing to take decisions about the next timeslot.

The last part (timeslot) of a message is not acknowledged within the above description, since timeslot n+2 will be free if in timeslot n is indicated that it is the last slot of this message. If necessary, the central transceiver must transmit extra information to acknowledge this last part.

The interleaved value I (I=2 in the above description) does not have to be two, but for the CAROLINE testsite, it is recommended to use I=2, because of its simplicity.

Emergency messages.

The message identity is not known by the interface board. However, an emergency message has to be recognized, because of its priority above normal messages. The emergency message is not allowed to be put in the normal toTRX buffer, because its delay might become too great.

A special emergency buffer has to be used. The PC has to tell the interface board that the message is an emergency message, so it can be put in the emergency buffer. Because the interface board does not know the start and end of a message, the easiest way that the PC can tell the interface board that the message is an emergency message, is done by means of an extra external pin from PC to interface board.

The data in the emergency buffer has priority above the normal toTRX buffer. This buffer has to be handled in a special way. The data out of this buffer is allowed to be sent also within a reserved timeslot and the mean delay before (re)transmission is much smaller than the "normal" mean delay.

If the random generator is implemented in the PC, there is no real need for a special emergency buffer. Then, all data is handled as emergency data, i.e., sending the data as soon as it is in the toTRX buffer and if one timeslot collides, the complete message will be resent (see BS acknowledgment). So there will never be a big delay, because there is at most one message in the toTRX buffer.
Conclusions

7. Conclusions.

In theory a multiple access protocol, which has a rather good performance for the CAROLINE application has been found. A choice has been made for the Reservation ALOHA protocol with some new ideas, which can be used in the CAROLINE application:

- acknowledgment is done by means of a Reservation (section 3.3),
- the protocol can rather easily be controlled by means of a limitation of the maximum number of retransmissions (section 3.3),
- the delay for emergency messages can be minimized by means of also accessing in the reserved timeslots with a higher access probability (section 3.5).

This found protocol has a high throughput (≥ 80%, depending on message length, section 3.4), it has a small delay for emergency messages, it is suitable for a large number of users, it is anonymous and the overhead needed in the downlink is small.

The interface board is able to perform the conversion from the NRZ coded RS232 lines of the PC to the synchronous FM0 coded lines of the ETACS transceiver. There arose some problems because the SCC used is a sophisticated peripheral which supports a lot of standards, but the CAROLINE protocol is not a standard (for example no CRC in the downlink). The implemented interface PC-link has a 19200 baudrate, the implemented interface transceiver-link has a 2400 baudrate. Because of lack of processing power in the Z80 μ-processor, it is very difficult to handle a 9600 baudrate for both the PC-link and the transceiver-link.

A slotted ALOHA protocol, without making use of acknowledgment, i.e., no automatic repeat request, has been implemented. The overhead in the downlink is rather small, but the overhead in the uplink is rather large due to the implementation of the protocol, when making use of the ETACS equipment.

The downlink consists of a timestructure which is needed to implement the protocol. Within GSM there already exists a timestructure. The overhead to make this timestructure in the ETACS system is rather small. Furthermore, just a couple of bits in the downlink are needed to control the protocol. The maximum implemented downlink data efficiency is 87%.

The overhead in the implemented uplink is very high. The uplink efficiency is just 54% for three reasons. The main reason being the slow switching transceivers. Within GSM the transceivers can switch much quicker and there will not be a data efficiency decrease because of this. Another reason for this low efficiency is the required communication overhead to give the preamble, postamble and CRC of the uplink timeslot within the ETACS system. Within GSM, this communication overhead is already performed. Finally, if the data length within one timeslot is made longer (i.e., greater than 64 bits), then the uplink efficiency will increase.
Conclusions

To get a baudrate which is higher than 9600 baud, while still using the ETACS system, another coding technique than FM0 has to be used. Furthermore, probably another, more powerful processor has to be used to handle this higher baudrate and to implement the Reservation ALOHA protocol completely. Using the ETACS transceivers at 19200 baud, the downlink efficiency will still be 87%, but the uplink efficiency will become 35%.
References

The cellnet cellular radio network. Part I - General system overview.

MATS-E, an advanced 900 MHz cellular radio telephone system: description, performance, evaluation, and field measurements.

Computer Networks.

[4] Abramson, A.
The ALOHA system - Another alternative for computer communications.

Closed loop stability controls for S-ALOHA satellite communications.
Symp. (Snowbird, UT), pp.2-10 - 2-19, September 1977.

Optimal retransmission control of slotted ALOHA systems.

Stability of a slotted ALOHA system with capture effect.

[8] Raychaudhuri, D.
Announced retransmission random access protocols.

Packet switching in radio channels: Part III - Polling and (dynamic) split-channel reservation multiple access.

[10] Binder, R.
A dynamic packet switching system for satellite broadcast channels.
References

General purpose packet satellite networks.

A system for broadcast communication: Reservation ALOHA.

Dynamic allocation of satellite capacity through packet reservation.

A new class of protocols for multiple access in satellite networks.

The SRUC protocol for satellite packet communication - a performance analysis.

[16] Rappaport, S.S.
Demand assigned multiple access systems using collision type request channels - Traffic capacity comparisons.

[17] Olie, Y. et al.
Throughput analysis of reservation protocols with tree-type reservation channel.

[18] Szpankowski, W.
Analysis and stability considerations in a reservation multiaccess system

[19] Rubin, I.
Access-control disciplines for multiaccess communication channels: Reservation and TDMA schemes.

Tree algorithms for packet broadcast channels.
References

[21] Murase, A. & Imamura, K.
Idle-signal casting multiple access with collision detection (ICMA-CD) for land mobile radio.

[22] Kleinrock, L. & Tobagi, F.A.
Packet switching in radio channels: Part I - Carrier sense multiple-access modes and their throughput-delay characteristics.

[23] Brewster, F.L. & Glass, A.M.
Throughput analysis of non-persistent and slotted non-persistent CSMA/CA protocols.


Microcomputer systeemarchitectuur.
Uitgeverij Stubeg Hoogezaand, 1986.


Appendix A. Circuit diagram of the interface board

- 8x4k pull-up resistors not used on lab version. Change to 74HC541 when used.
Appendix B

Appendix B. Program development with AZTEC C80/ROM

The software conversion steps from C to Intel hex format are shown in fig. A.1.

*step 1*
- Editor
- C source file

*step 2*
- Compiler
- 8080 source file

*step 3*
- Optimizer
- Z80 source file

*step 4*
- Assembler
- Object module
- Librarian

*step 5*
- Linker
- Subroutine library
- Executable code

*step 6*
- Hex80
- Hex code

*fig. A.1 Program development with Aztec C80/ROM.*
step 1: Creating the source program.
For creating the C program, one can make use of any type of editor. We used the Turbo-C editor. This has the benefit that one can at this stage check on compiler errors and - warnings.

step 2: Compiling to 8080 assembler code.
This step creates the 8080 assembler code. This code can be used on a Z80 µ-processor. But because the Z80 is an extension of the 8080 µ-processor, this 8080 assembler code can be optimized.

step 3: Optimizer to Z80 code.
With this step the program code will be optimized up to about a factor 0.6.

step 4: Assembler.
This step will create the object code of the program.

step 5: Linker.
The object code version of the program must be linked to functions that are in the library. We tried to make as little use as possible of library functions, because we do have no control over these functions. With the linker it is possible to generate a symbol table, which can be used during step 6. The format of this symbol table is first the address, then the symbol. The format of the symbol table the SDT816 emulator uses, however, is first the symbol, then the address. This symbol-address conversion has to be done after step 6.

step 6: Convert to Intel hex records.
The last step to be performed is the conversion to the Intel hex records which can be downloaded into the emulator. By default, one or more files are generated, each of which contains hex records for one 2k-byte section. However, note that such a 2k-byte section includes the overhead because of the representation of the Intel format. Said differently: by default, the program size within one section will be less then 2k-byte. We increased the section size to 8k-byte, because of the 8k-byte RAM.

The software was written in the C-language, but because of the Z80 µ-processor, the debugging had to be done in Z80-assembly language. The emulator performs the disassembling from the Intel hex code to the Z80 assembly language. To perform this disassembling, it makes use of the symbol table. Function calls and global variables will become symbols. Having many symbols will aid understanding of what the cross-compiler has done. That is why the C code has small functions and many global variables to make debugging in Z80 assembler easier.
Appendix C. Calculation of the number of zero-bit insertions.

Problem:
A bitstream of length N is declared as an N-string. Suppose that the probability of a "1" bit is equal to the probability of a "0" bit occurring, i.e., 1/2. In an N-string there can be zero-bit insertion after five consecutive "1s". The probability, \( P(N,k) \), that in an N-string k "0s" will be inserted will be calculated.

Solution:
Having an N-string with k "0s" inserted will mean there are k groups of five consecutive "1s". This means:

\[ P(N,k) = 0 \quad \text{if} \quad N < 5k \]  
\[ P(5k,k) = (1/2)^{5k} \]

Having a 5k-string with k "0s" inserted will happen only if the string consists of all "1s". This means:

Having an N-string with \( N \leq 4 \) implies no "0"-bit will be inserted, and having an N-string with \( N = 5 \) implies that there will only be a "0"-bit insertion if the string consists of all "1s". This means:

\[ P(1,0) = P(2,0) = P(3,0) = P(4,0) = 1 \quad ; \quad P(5,0) = 31/32 \]

Now considering N-strings with \( N \geq 6 \) and \( k \geq 0 \), the contribution to \( P(N,k) \) of strings of different lengths will be calculated. The probability that an N-string starts with a "0" is 1/2. The probability that in an N-string, which starts with a "0", k "0s" will be inserted is \( (1/2)P(N-1,k) \). So, the contribution to \( P(N,k) \) of an N-string, which starts with "10", "110", "1110", and "11110" is \( (1/4)P(N-2,k) \), \( (1/8)P(N-3,k) \), \( (1/16)P(N-4,k) \), and \( (1/32)P(N-5,k) \) respectively. The contribution to \( P(N,k) \) of all remaining strings, i.e., strings starting with "11111", is \( (1/32)P(N-5,k-1) \).

\[ \{=0 \quad \text{if} \quad k=0 \} \]

Therefore, \( P(N,k) \) can be expressed by means of the following recursive expression:

\[ P(N,k) = \sum_{i=1}^{5} \frac{1}{2^i} P(N-i,k) + \frac{1}{32} P(N-5,k-1) \]

In expression (4), \( P(j,-1) = 0 \) for all j.

By means of expressions (1) to (4), \( P(N,k) \) can be calculated: firstly, calculate by means of (3) & (4) \( P(j,0) \) for \( j \leq N \); secondly, if \( P(j,L) \) is known for \( j \leq N \) and \( L \leq k-1 \), \( P(j,L+1) \) for \( j \leq N \) can be calculated by means of (1), (2), and (4). In table C.1 and fig. C.1 the calculated results for \( P(80,k) \) are shown.
# Appendix C

*Table C.1. Calculations of $P(80,k)$.*

<table>
<thead>
<tr>
<th>$k$</th>
<th>$P(80,k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$2.677\times10^{-1}$</td>
</tr>
<tr>
<td>1</td>
<td>$3.758\times10^{-1}$</td>
</tr>
<tr>
<td>2</td>
<td>$2.385\times10^{-1}$</td>
</tr>
<tr>
<td>3</td>
<td>$9.048\times10^{-2}$</td>
</tr>
<tr>
<td>4</td>
<td>$2.283\times10^{-2}$</td>
</tr>
<tr>
<td>5</td>
<td>$4.041\times10^{-3}$</td>
</tr>
<tr>
<td>6</td>
<td>$5.146\times10^{-4}$</td>
</tr>
<tr>
<td>7</td>
<td>$4.764\times10^{-5}$</td>
</tr>
<tr>
<td>8</td>
<td>$3.200\times10^{-6}$</td>
</tr>
<tr>
<td>9</td>
<td>$1.539\times10^{-7}$</td>
</tr>
<tr>
<td>10</td>
<td>$5.156\times10^{-9}$</td>
</tr>
<tr>
<td>11</td>
<td>$1.153\times10^{-10}$</td>
</tr>
<tr>
<td>12</td>
<td>$1.601\times10^{-12}$</td>
</tr>
<tr>
<td>13</td>
<td>$1.227\times10^{-14}$</td>
</tr>
<tr>
<td>14</td>
<td>$4.144\times10^{-17}$</td>
</tr>
<tr>
<td>15</td>
<td>$3.680\times10^{-20}$</td>
</tr>
<tr>
<td>16</td>
<td>$8.272\times10^{-25}$</td>
</tr>
</tbody>
</table>

*fig. C.1 Probability of having $k$ "0"-bit insertions in an 80 bit string.*
Appendix D

Appendix D. Abbreviations

AVL  Automatic Vehicle Location
BSC  Base Station Control
BSS  Base Station System
BS(s) Base Station(s)
BTS  Base Transmitter Station
CARIN CAR Information & Navigation system
CAROLINE CAR to ROad Link for Information, Navigation and Efficiency
CCITT Comité Consultatif International de Télégraphique et Téléphonique
CDMA Code Division Multiple Access
CRC  Cyclic Redundancy Check
CSMA Carrier Sense Multiple Access
CTS  Clear To Send
DTR  Data Terminal Ready
EOM  End Of Message
ETACS Enhanced Total Access Communication System
FDMA Frequency Division Multiple Access
FM0  biphase mark coding technique
GSM  Group Special Mobile
ICMA Idle-signal Casting Multiple Access
IE  Interrupt Enable
MSC  Mobile Station Control
MSS  Mobile Station System
MS(s) Mobile Station(s)
MTS  Mobile Transmitter Station
NRZ  Non Return to Zero
.OSI Open Systems Interconnection
PREVU PRoduct for Expected Vehicle Usage
PSTN Public Switched Telephone Network
RDS-TMC Radio Data System - Traffic Message Channel
RTS  Request To Send
SCC  Serial Communication Controller
SDLC Synchronous Data Link Control
SSMA Spread Spectrum Multiple Access
TDMA Time Division Multiple Access
TRX  transceiver(s)
WR  Write Register