A distributed, mobile positioning system for wireless, handheld devices

by

Fredrik Christiansson

A thesis project at lesswire AG
under supervision of Prof. G. Q. Maguire Jr.,
Royal Institute of Technology, Stockholm Sweden,
in fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Computer Science

Approved:

____

Prof. G. Q. Maguire Jr.

June 2000

Frankfurt an der Oder, Germany

Keyword list

Location awareness, positioning, mobile devices, wireless, handheld, fares, Bluetooth, time difference of arrival, TDOA, synchronization, timing, network, picocell.

Abstract

This thesis investigates the possibilities of implementing a location awareness mechanism for the so-called lesswire localNavigator. The author claims that it is possible to implement such a mechanism within the given prerequisites and constraints, even though with today's technology it may not be economically feasible.

Due to the lesswire's constraints: high accuracy (67%), high-resolution (12 m^2) and no hardware modification allowed to the mobile device, the suggested scheme uses Time Difference Of Arrival technology (TDOA).

The main *advantage* of TDOA, as stated in this thesis, is the fact that it is almost totally independent of the preferred wireless technology of the mobile device. TDOA technology therefore, can be applied to a wide range of wireless networks (primarily TDMA, CDMA, FDMA – based). A *disadvantage* of this scheme is the fact that the network infrastructure needs to be extremely well synchronized - which in turn implies higher costs.

Depending on how the synchronization problem is solved, the proposed system may well be economically feasible in the near future.

Acknowledgement

I wish to thank Prof. Dr. Rolf Kraemer for giving me the opportunity to work under his supervision, together with a lovely team of international co-workers at lesswire – no one mentioned and no one forgotten! Thanks to lesswire, a desire of coming to Germany, experience its people, culture and language was fulfilled. I am also very glad it happened to be lesswire and Frankfurt an der Oder in particular, considering the many opportunities at hand before everything was set.

Also, I wish to express gratitude to Prof. G. Q. Maguire Jr. and Mr. Gunter Schoof, who both have supported me with a great deal of care, valuable knowledge and references from the very start to the very end.

Not to forget is Bengt Sjöberg for his supportive feedback and patient assistance when working with the manuscript, as well as his great sense of humor which made me carry on in moments of despair and irresolution.

Finally, I would like to thank my mother, who, like no one else, has supported me in good times and bad times, in joy and in sorrow. Her love, care and positive thinking have given me the necessary strength and confidence to fulfill my missions – whatever they might have concerned.

Contents

List of figures

1. Introduction

The need for providing position data to a potential user of lesswire's product is obvious: it opens up a wide range of new services and applications, as we will see in this introductory chapter. The wireless world itself is mobile by nature, and the absence of a fixed structure in which it's obvious at any instant where a user is located, requires a new way of thinking not only in terms of system design, but also in terms of what new services can be provided.

This chapter will give the reader a brief overview of lesswire – their business profile and incentives for incorporating local awareness capability in their product line. In addition, different techniques of solving the positioning problem in a more general, scientific way will be outlined. Finally a brief discussion of the results and work, which will be presented in this thesis, is given.

1.1. Incentives for location awareness

1.1.1. Background of this work

The information technology market is focusing more and more on wireless products and applications. This is true for both the consumer market and industry: As evidenced by the fast growth of mobile telephones and consumer demand for more and more services. Producers seem willing to meet this demand and the current positive trend, among investors in IT related businesses, creates a good environment for new technology to evolve.

The Internet has so far largely been based on fixed infrastructure technologies, although there have been notable efforts with mobile packet data [1] and Mobile-IP [2]. Wireless communication for the broad market has primarily evolved by means of the mobile telephone industry. An interesting thing would therefore be to create "The Wireless World" in which people (or other objects) communicate and have access to the entire Internet - or other nets as well without any requirements on workspaces, power cords, network cables, etc. This, of course, does not necessarily have to be achieved by mobile telephones, although these devices are the most common handheld devices for wireless communication today.

My own focus on telecommunication, based on my acquaintance with Prof. Gerald Q. Maguire Jr. at the Royal Institute of Technology and his contacts with companies who share the above stated visions, brought me to lesswire AG [1] in Frankfurt an der Oder, Germany.

[1] See http://www.lesswire.de

Their first generation wireless, handheld devices (and corresponding infrastructure) is currently under development (April 2000). Their intended use is primarily for at big fairs, exhibitions or other similar events, where visitors may want to access all kinds of data, such as: their current position, directions to a certain stand, directions to other people, access to information, ... and even the ability to save information easily about the exhibitors, exhibits, etc.

This thesis project deals exclusively with the positioning problem, i.e., knowing where a user is relative to one or more elements of the infrastructure.

1.1.2. lesswire – a short presentation

The vision of the lesswire team, as described briefly in the previous section, is to provide a user, equipped with some sort of a mobile client, relevant information as a function of the user's position and/or profile.

Typical applications are at fairs, airports, shopping malls, etc. The user may want to get access to information about a specific exhibitor at fairs without having to stand in a line for a long time or carrying several pounds of brochures, specifications, etc. The award winning localNavigator (as the entire system is called) is the intelligent interface between the user and the local information provider. It consists of three major components:

Client:	A browser plug-in providing client software for mobile wireless terminal.
Base station:	A hardware interface between the mobile client and the platform that serves as a transceiver/receiver. This base station also implements the lower layer locator system and therefore is of great importance to this work.
Platform:	A system (-server) software that provides the locally relevant information.

The position mechanism is without any doubt, of great importance to this system. Position related issues that may arise are for instance:

- Where am I?
- How do I find him/her?
- Where do I find/how do I get to X, where X is any object (i.e. a set of coordinates to be more explicit), within the local environment?
- How far is it to X?
- The infrastructure may also want to know where a user is and send news and announcements.

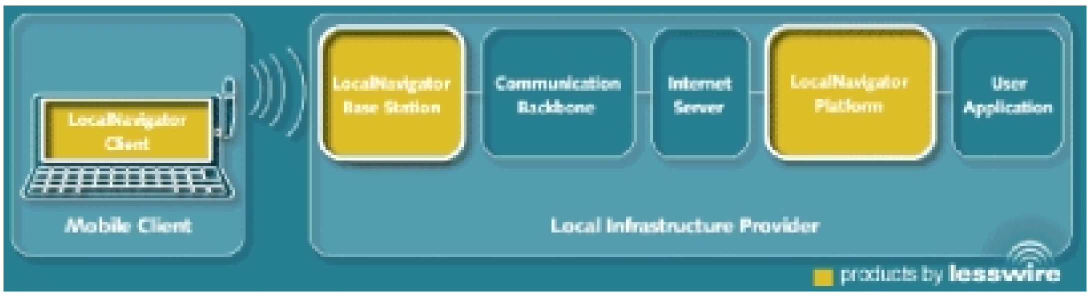


Figure 1: The localNavigator

As stated in the localNavigator specification, “the preferred wireless standard is Bluetooth”, but the base station will, for the time being, also support the IEEE 802.11b wireless LAN standard for data and IrDA or RC5 infrared for the transmission of location information.

This work is solely focused on a radio solution (with Bluetooth in mind), but the theoretical discussions on position calculation may well be applied to other media (infrared or sound for instance).

For a more thorough presentation of lesswire, see http://www.lesswire.de.

1.2. Constraints and prerequisites

To position objects in the real world is not a new thing. Even the broad end-user market has, to some extent, some knowledge of a system such as GPS (Global Positioning System) which was launched in the early 1960's [3]. So called *city navigators*[2] now are really becoming feasible even for smaller companies and organizations. What makes the needs at lesswire specific? There are primarily three demands, stated by the lesswire team, which optimally should be fulfilled and taken into account when approaching this problem. Every one of these demands must be considered and influence every decision made when proposing a system:

1. **High accuracy and high position resolution.**
2. **Affordability (low complexity and cost)**
3. **No modifications**. No *hardware* modifications to the user's device (from now on referred to as a PDA [Personal Digital Assistant]).

In addition, an obvious assumption that follows, is identification: the PDA must provide a means for identifying itself to the infrastructure through whatever protocols it may be using (preferably Bluetooth).

1.2.1. High accuracy and high position resolution

The high accuracy demand is evident, considering the actual application, but yet challenging – perhaps the most challenging of the above stated constraints. Since we are dealing with a speed-of-light domain, an error of one microsecond yields an error of about 300 meters (assuming the speed of a radio wave is approx. speed of light in vacuum – for exact values consult appropriate tables), which of course, is far too inaccurate in normal indoor environments.

[2] For instance http://www.citykey.com

Hence, we will have to work with time domains in the order of nanoseconds (ns). An accuracy of 67% is desirable; i.e. we want to give a correct position 67% of the times at a certain resolution.

High position resolution implies that the calculated area, in which the user is said to be in, is not allowed to be greater than x m^2. Accuracy and resolution are not equal and the author would like to stress this fact. Unfortunately, they are amazingly often used synonymously, or in other erroneous ways. So called cell-id system may claim that the user is inside a certain cell and it may do so with high accuracy, but the fact that the user is inside the cell may not be enough. Hence, producing a system with high accuracy, but low resolution.

A 12 m^2 resolution (which corresponds to a circle with a 2 meter radius) will be assumed in this text. A typical short-range radio picocell, used for example in Bluetooth, in short-range mode with a radius of 10 meter, results in a resolution of 100pi/6 ≈ 52 m^2, assuming complete antenna coverage (at least three base stations cover any single spot) and a cell-id based system. It is important not to mix up antenna coverage and desired resolution: a complete antenna coverage with +/- 10 meter Bluetooth cells does not necessarily imply a +/- 10 meter resolution. We still, as stated above, require +/- 2 meter resolution for the system to be practically useful.

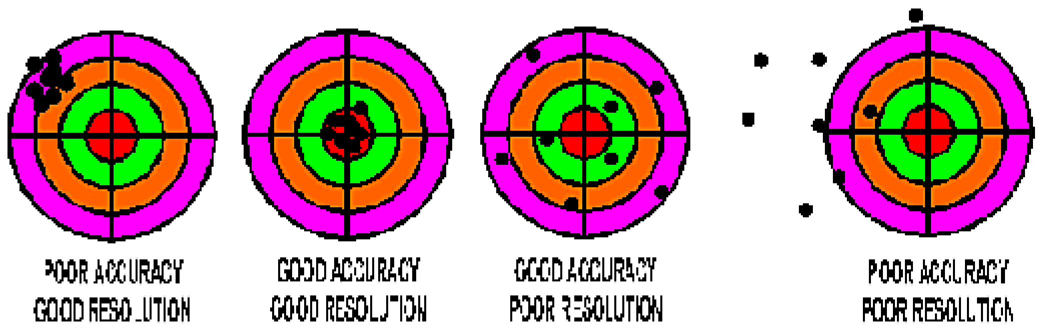


Figure 2: Resolution and accuracy

Compared to current commercial systems (most of them designed for outdoor purposes), 12 m^2 *is* extremely high resolution.

The Federal Communications Commission (FCC) requires that the position of a wireless 911 (US emergency number) caller can be determined within a radius of no more than 125 meters ($\approx$ 49 063 m^2) in 67 percent of all cases (starting October 1, 2001). For an outdoor environment, this may be enough, but it does not make sense for an indoor environment.

1.2.2. Affordability

To propose a system that theoretically meets all the constraints is a different thing from making it realistic in terms of implementation. As we will see, our system requires top-of-the line hardware, able to work in the GHz-range. This is true both for the synchronization and position problem. However, for a commercial system, there is a strong desire to keep complexity and costs down. Also, the fact that the number of base stations tends to be large (approx. 2000 per 100 000 m^2, where position capabilities are desired for the standard Bluetooth base stations) forces us to propose something “cheap **and** good”.

1.2.3. No modifications

This third demand really makes this work challenging. Several systems with ActiveBadge-alike [4, 5] systems have been proposed, but physically adding a badge-like, external component to the PDA is considered a hardware modification and therefore violates constraint number three. The lesswire team stated that a software plug-in is necessary – not only for positioning – but also for other reasons. In such a software package, the necessary position mechanisms could be incorporated.

An obvious assumption, but still important to state: we have to know to whom we shall send location aware information and also from who we receive it. Bluetooth supports addressing, a fact we can benefit from. The solution proposed in this paper assumes that the PDA acts as a master and by transmitting broadcast messages, the base stations will be able to measure the arrival time.

The proposal for an implementation described in this paper, is not necessarily bound to a specific protocol, but the lesswire team has focused on Bluetooth [6] technology, and we will therefore assume Bluetooth as the protocol of preference. Bluetooth is basically a proposed open standard for cordless communication between a variety of devices such as mobile telephones, keyboards, mice etc. It’s the result of work between Nokia, Toshiba, Ericsson, Motorola, and Intel.

1.2.4. Thesis outline

The subsequent report will be structured as follows: chapter 1 *(Introduction)* discusses the major incentive and background for position location at lesswire and also various techniques for incorporating location awareness. Finally a brief discussion of why TDOA is chosen for this application. Chapter 2 *(Different position techniques)* describes different position techniques – advantages and disadvantages. Chapter 3 *(Position calculation with TDOA)* describes TDOA in more depth. Chapter 4 *(Simulation and error analysis)* deals with an error analysis by comparing two algorithms and performing simulations. Chapter 5 *(A proposed system design)* is the subject of a proposed system design for lesswire. Chapter 6 *(Discussion and future work)* presents the general results and concludes the thesis in terms of results and conclusions.

2. Different position techniques

2.1. Introduction

Many different approaches have been proposed from various sources over the years. In addition, ever since the FCC mandate[3], competitors in the mobile telephone market have found an even stronger incentive for position solutions. Accuracy and feasibility (i.e. keeping costs low) are the most important considerations. lesswire probably can learn a lot from them – especially the field trials [7].

We can roughly divide these approaches into two categories: those that require modifications to the PDA and those in which all processing is done in the infrastructure (basically everything except for the PDA, e.g. base stations, network etc.). However, this first category can be immediately omitted, since it violates constraint three (on page 5).

[3] See http://www.fcc.gov/e911/

2.1.1. Techniques not requiring changes to the PDA

Basically three different schemes exist, although hybrids of them exist which makes it possible to improve accuracy: AOA (Angle Of Arrival), TOA (Time Of Arrival), and TDOA (Time Difference Of Arrival). In this section, they will briefly be introduced from different aspects such as sensitivity to errors and complexity (which often implies higher costs).

2.1.1.1. AOA – Angle Of Arrival

By using the geometrical axiom stating that the point of intersection between two straight lines is unique, we can calculate the position of the PDA. If these straight lines correspond to the shortest distance between the PDA and the base station (at least two base stations) we can calculate the point of their intersection by calculating the angles from a referential direction as proposed in Figure 3. This scheme requires at least two antennae per base station. To avoid phase ambiguity, they should be placed less than one wavelength apart. Most often, an *array* of typically 4 to 12 antennae is used [7]. A nice AOA-introduction can be found in [8].

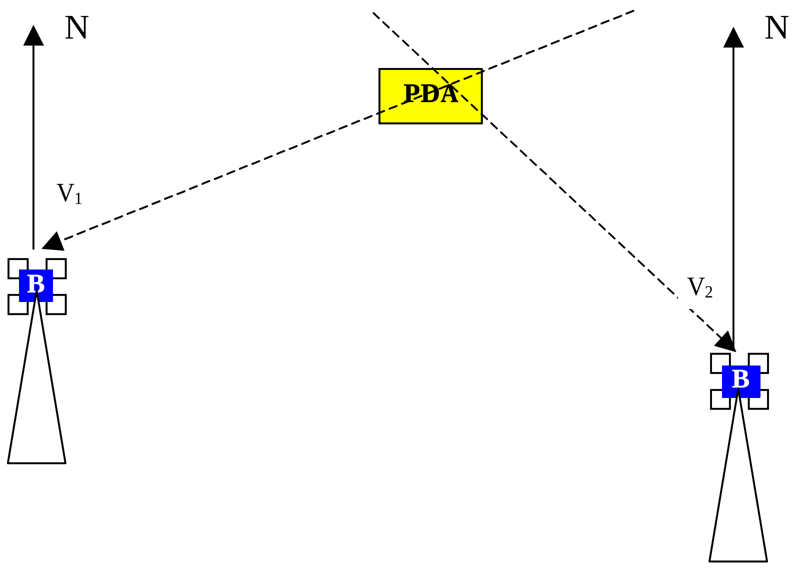


Figure 3: Principles of AOA position calculation

By measuring the time difference at each base station when the signal reaches each antenna, the angle can easily be calculated according to:

$$V_x = 90 \times f \times \Delta t$$

Eq. 1

where *f* equals the frequency of the incoming signal, Δt the measured time difference between the antennae at one base station. The transformation to x-coordinates (y-coordinate not shown here) of the PDA is given by:

$$X = \frac{x_2 \times \tan(90 - v_2) - x_1 \times \tan(90 - v_1) + y_2 - y_1}{\tan(90 - v_2) - \tan(90 - v_1)}$$

Eq. 2

where *x* and *y* correspond to the absolute coordinates of the base stations and *v* to the angle. The indices refer to base station 1 or 2.

An advantage with the AOA approach is that only two base stations are required. Drawbacks are the complex structure of the antenna array, the sensitivity for multi-path signals (since they affect the angle). In addition, AOA tends to be very sensitive to errors, especially when the PDA is far away from the base stations known as the GDOP (Geometric Dilution Of Precision) problem. Critical points are also the ones close to equal angles. This fact can be observed in Figure 4, which shows the x-coordinate of an assumed PDA. Base stations are 300 meters apart in this example.

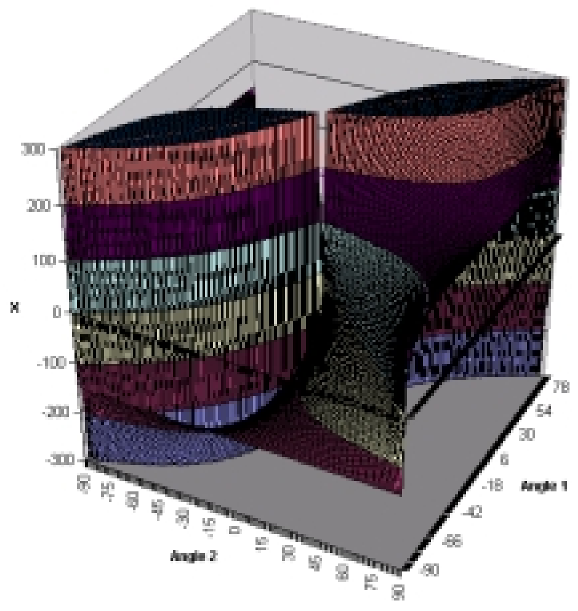


Figure 4: AOA with X = f(angle1, angle2)

The two ellipse-shaped midsections clearly define the critical set of coordinates.

The fact that AOA technology relies on antenna arrays, ordinary one-antennae-per-base-station infrastructure will not suffice and an overlay structure will be required. Sensitivity to multipath signals and new infrastructure doesn't make AOA itself a realistic alternative for lesswire. However, it may be used in combination with other techniques, like for instance TDOA[4].

2.1.1.2. *TOA - Time Of Arrival*

An alternative approach is to basically initiate a time-packet request to the PDA, wait until the packet returns and from this total time, and calculate the time it takes for the signal to reach the PDA. This is a neat and straightforward way to achieve what we are looking for, but as we will see, it has some drawbacks.

[4] Animated demo of a hybrid system can be seen at SigmaOne's site (http://www.sigma.com/tdoa.htm)

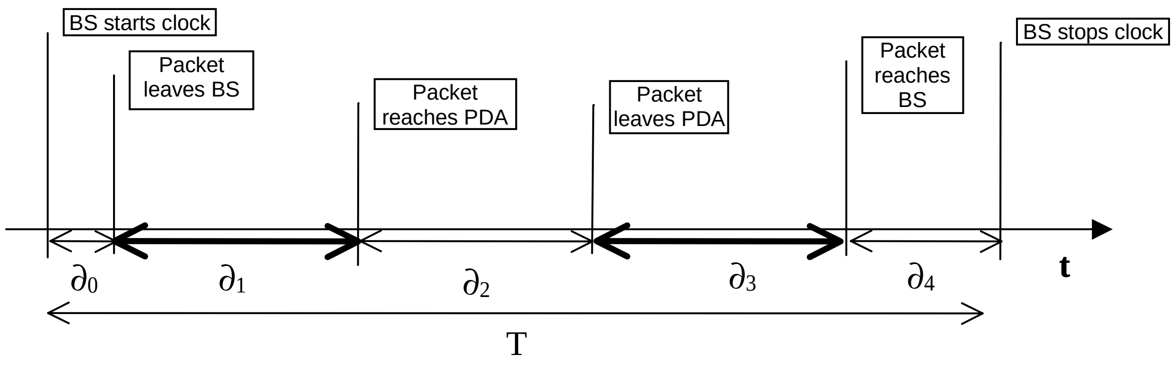


Figure 5: TOA timing diagram

$$T = \partial_0 + \partial_1 + \partial_2 + \partial_3 + \partial_4$$

Eq. 3

Assuming that the signal needs equal time to travel to and from the PDA. We therefore can substitute ∂_3 for ∂_1, which eliminates ∂_1:

$$\partial_1 = \partial_3 = \frac{T - \partial_4 - \partial_2 - \partial_0}{2}$$

Eq. 4

Time can easily be translated to distance and by repeating this for at least three base stations, then we can, by means of triangulation, calculate the position. The main drawback with this method is the fact that the processing delays ($\partial_{0,2,4}$) are most likely to vary due to different PDAs produced by different manufacturers. The fact that we do not know enough about the processing variances for the different PDAs entering our infrastructure makes this scheme impossible, even though we could get extremely high accuracy and carefully predicted delays at the base station side. A very appealing conclusion though, is that we do not have to care about synchronization. The only time constraint is that the internal time domain at each base station must not drift too much. No synchronization is needed between the base stations, nor between any of the base stations and the PDA.

2.1.1.3. *TDOA – Time Difference Of Arrival*

As its name suggests, this technique calculates the time difference that occurs between base stations when the same transmitted signal leaves the PDA and reaches the base stations within reach. To achieve this, the signal is recorded at a very well synchronized time interval and by different cross-correlation techniques the signal peaks can be detected and the time difference calculated [9].

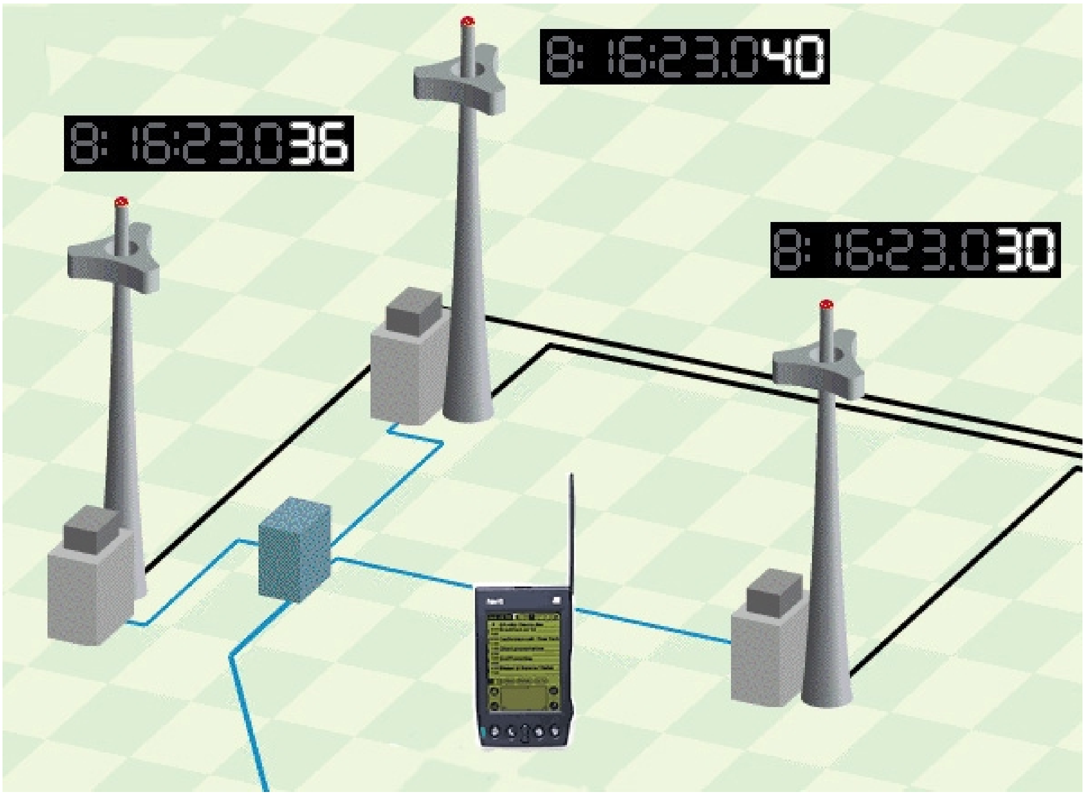


Figure 6: TDOA time difference scheme

Each pair of time differences defines a hyperbola between the corresponding base stations – a hyperbola on which the PDA may exist. By repeating the scheme for three base stations and their combinations of time differences, the set of hyperbolas together defines a unique point of intersection.

The most appealing feature of the TDOA scheme is the fact that *no* modifications at all are needed to the PDA, as everything is done in the infrastructure. Furthermore only one transmission, at least theoretically, is required, namely the one from the PDA to the base station. In practice though, the signal may be distorted and in the worst case, useless for the position calculations. That will force us to request re-transmissions.

What is more useful is, we don't need to know anything about the absolute transmission time of the PDA and no synchronization is needed between PDA and infrastructure. Normal antennas can be used and also, since a time *difference* scheme is used, multipath signals can be cancelled out because the time difference of arrival of the first signal remains the same. This is not the case for AOA where the angle itself will be wrong or for TOA, where no such automatic cancellation can be performed.

Major drawbacks are, as mentioned earlier, the nanosecond accuracy that has to be implemented at infrastructure level, but also some considerations when it comes to increased computing power and network traffic.

Despite that, this work indicates that TDOA is the technique of choice, when considering the lesswire constraints.

More reading on the general position problem is to be found in [10, 11, 12, 13, 14]

2.2. Research outline

2.2.1. Purpose of this research

The intention of this research is to propose an appropriate solution to the location problem. This in turn will incorporate an algebraic approach for the position calculation and also a system structure for l e s s w i r e to use for implementation purposes. Also, there will be an error analysis to get a feel for where to put the effort to reduce errors.

3. Position calculation with TDOA

3.1. Introduction

This chapter is devoted to the algebraic aspects of the TDOA position scheme. To agree upon and understand the basic algebra in terms of the hyperbolic formulas is not the main issue. The challenge is rather to choose appropriate algorithms to solve the nonlinear hyperbolic equations. Unfortunately, the algebraic considerations must not be considered in isolation from the actual application: an environment with fast-moving PDAs require fast algorithms in which a trade-off between speed and accuracy has to be done. In addition, an algorithm demanding too much system resources in terms of computing power is undesirable.

The synchronization among the participating base stations is an unquestionable constraint for implementing a TDOA system. In this paper though, accurate synchronization is assumed and no concrete system solution for achieving this is proposed. Still though, a brief discussion will deal with this topic due to its importance for the system implementation.

3.2. The time synchronization problem

As mentioned earlier, this paper does not claim to treat the synchronization problem to its full extent, but rather explains the need for synchronization, what resolution is desired, and we provide references to earlier work and possible solutions. No doubt, the major drawback with TDOA estimation is the required synchronization between the participating base stations. Depending on the size of the network (i.e. the costs) and the required resolution, a distributed or centralized solution can be used. It is important to stress that the TDOA algorithm does *not* require a universal, global synchronized time base (UTC or similar). The important thing is only that the relevant base stations are synchronized relative to each other.

The error budget, when it comes to the timing, is treated in *Simulation and error analysis* on page 55 and primarily we have two sources of errors: the accuracy in synchronizing and the accuracy in triggering the clocks. Summing up the two sources, they must not exceed 8 ns. This gives us a hint of how good the synchronization must be, but it is impossible at this point to exactly specify a harder limit – choice of hardware, how well the triggering mechanism performs and further field studies will finally set this limit. We will from now on assume it is somewhere “below 8 ns”.

3.2.1. Distributed, algorithmic solutions

In this section we briefly discuss a general approach to the synchronization problem, followed by a Bluetooth specific one, due to the fact that Bluetooth provides synchronization by using its, for each and every packet mandatory, access code. This is further discussed in the baseband specification, section 4.2 in [6].

3.2.1.1. General solutions

Ever since Lamport's pioneering work in the seventies, introducing a formal approach to the problem of ordering events in a distributed environment, a great effort into this area has been made, and among these the work of D. L. Mills should be stressed [15, 16, 17]. Mills claims [18] that an accuracy equal to, or better than 10 ns is possible in a 100 Mb/s Ethernet, but that additional hardware may have to be added. His effort in [19] is a serious move into the nanosecond time domain and provides further improvements of [15]. The basic idea is to replace the clock-synchronizing daemon in the OS kernel, which is done purely in software.

The NTP (Network Time Protocol) constitutes the basis for Mill's work and it is the common standard for the Internet community. As such, it provides accuracy on the order of milliseconds, but for the purpose of our system design, we assume a typical LAN environment and the fact that we (i.e. lesswire itself) can do necessary hardware assignments, this significantly improves our capability of reaching the desired nanosecond accuracy.

Is important to be aware of the fact that active components like routers and switches contribute to unpredictable time delays. The bigger the LAN (in which our infrastructure is communicating), the higher the probability for unwanted introduction of time jitter, since more and more active components are being introduced.

In addition, the work of Eidson et al. [20] is an interesting approach to the synchronizing problem. They suggest a non-hierarchical scheme in which any node may initiate a synchronization. Significant to their invention is the usage of a Time Packet Detector (TPD) that detects and recognizes timing data packets. They specifically address and treat the timing jitter problem caused by the protocol stack and operating system that, for instance, neither Kopetz [21] nor Hosgood [22] have managed to overcome.

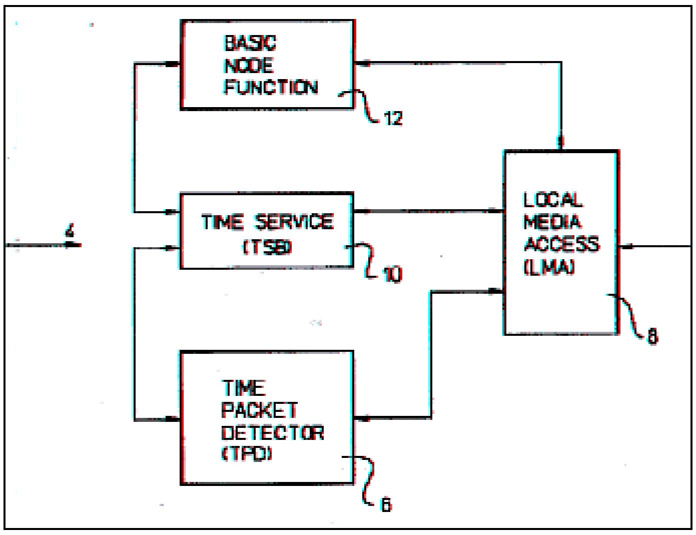


Figure 7: System overview of the Eidson patent

Their effort at Agilent Technologies[5] has focused on synchronization and their field trials show that their algorithm together with the necessary hardware can provide synchronization accuracy below 200 ns in a typical Ethernet LAN environment. This is a significant improvement compared to dominating time standards for Ethernet LANs, but would not be sufficient for our needs (below 8 ns). A worst-case deviation of 200 ns alone would contribute to an error of approx. 60 meters.

[5] See also http://www.tm.agilent.com/tmo/press/English/PRTM0419901.html

3.2.1.2. A Phase Locked Loop (PLL) solution

A common way to achieve a stable, local clock is by using a PLL. The PLL basically take a reference voltage as input and produces a stable frequency. In other words: frequency as a function of input voltage.

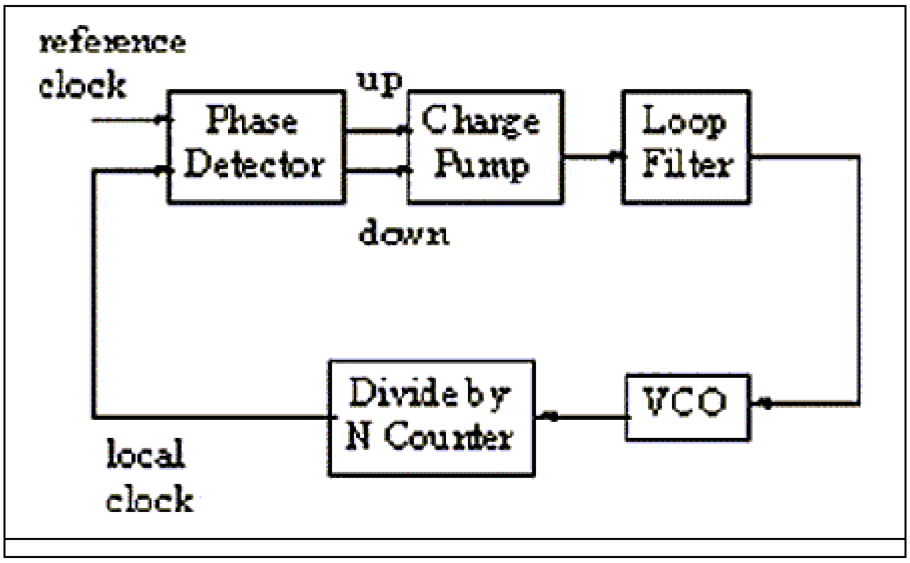


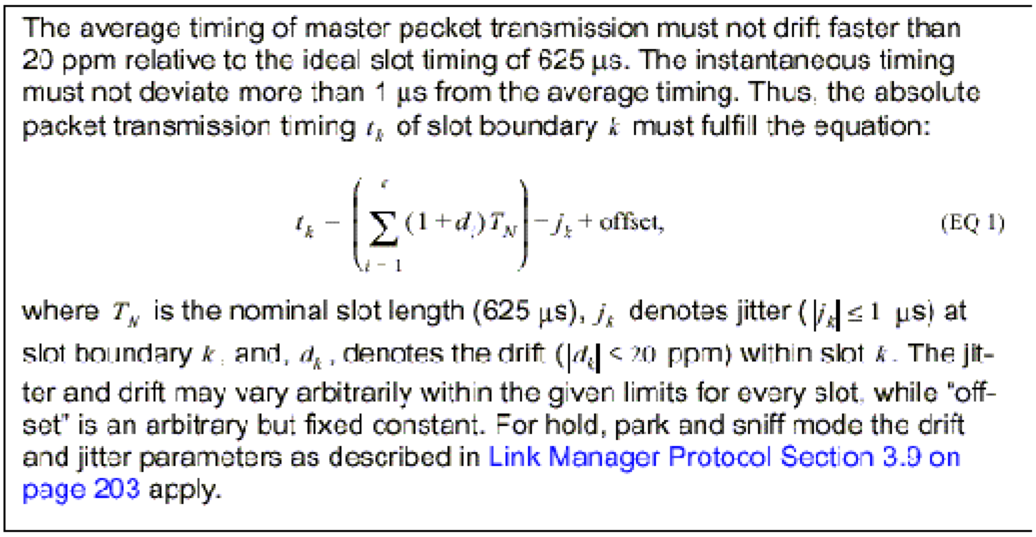
Figure 8: Possible PLL configuration

[23] believes such a PLL could be “completely integrated in CMOS technology” together with the necessary logic (counter etc) and also provide the resolution we need.

3.2.1.3. A Bluetooth synchronization approach

The most obvious method of all would probably be to use the built-in synchronization of Bluetooth. *If* Bluetooth could provide reliable synchronization down to the nanosecond, implying that receiver (i.e. base station) as well transmitter (PDA) are synchronized, we could easily implement the TOA scheme described in section by simply timestamping the packets.

Unfortunately the Bluetooth synchronization operates in the microsecond domain [6] (see page 22) rather than the nanosecond domain – hence it is roughly an order of magnitude worse than that of Eidson and three orders of magnitude of magnitude worse than we need.



The average timing of master packet transmission must not drift faster than 20 ppm relative to the ideal slot timing of 625 µs. The instantaneous timing must not deviate more than 1 µs from the average timing. Thus, the absolute packet transmission timing t_k of slot boundary k must fulfill the equation:

$$t_k - \left(\sum_{i=1}^{k}(1+d_i)T_N\right) - j_k + \text{offset}, \qquad \text{(EQ 1)}$$

where T_N is the nominal slot length (625 µs), j_k denotes jitter ($|j_k| \leq 1$ µs) at slot boundary k, and, d_k, denotes the drift ($|d_k| \leq 20$ ppm) within slot k. The jitter and drift may vary arbitrarily within the given limits for every slot, while "offset" is an arbitrary but fixed constant. For hold, park and sniff mode the drift and jitter parameters as described in Link Manager Protocol Section 3.9 on page 203 apply.

Figure 9: Timing in Bluetooth (from BT spec. 1.0 B)

The timing accuracy in the Bluetooth standard may be sufficient for synchronous (max 64 kb/s) and asynchronous (max 721 kb/s) communication, but it is not sufficient for our application. The above quote from the Bluetooth specification basically means that the allowed jitter *alone* is enough to cause an error of 1 µs, which would translate into an approx. 300 meters of error. That is unacceptable. G. Fischer also [24] expressed his doubts that the built-in Bluetooth synchronization is good enough[6].

[6] Dr. Gunter Fischer is currently (March 2000) in charge of a pioneering Bluetooth project at IHP dealing with system design of a completely integrated Bluetooth chip in CMOS technology.

3.2.1.4. A GPS solution

A common way to solve the synchronization issue is by using the worldwide available GPS system, which not only provides position information, but also accurate time. A possible implementation typically equips each base station with a GPS receiver. The GPS has two drawbacks for our application: first of all, it requires line of sight to the transmitting satellites. This could be solved by mounting an outdoor antenna and connect it to the infrastructure via cabling. However, we must make sure that the cabling does not introduce too much errors in terms of for instance thermal heating. Materials like optical fibers, with very low thermal expansion coefficients.

With GPS today, we can obtain a timing accuracy of approx. 50 ns[7], which will not do for our application. The so-called Differential GPS (DGPS) does *not* improve time accuracy (only position accuracy).

3.2.2. A centralized solution

A straightforward approach to the problem is to have a central, common time base, which is physically connected to each and every participating base station. This solution resembles of the GPS solution described in the previous section. Time delays in the connecting media (typically cables) must be taken into account.

[7] See for instance http://www.trimble.com/products/catalog/oem/aceutc.htm

In comparison to the earlier mentioned distributed solution and that used for the Internet community, such a centralized solution is possible for our application, as long as we are solely considering an indoor environment.

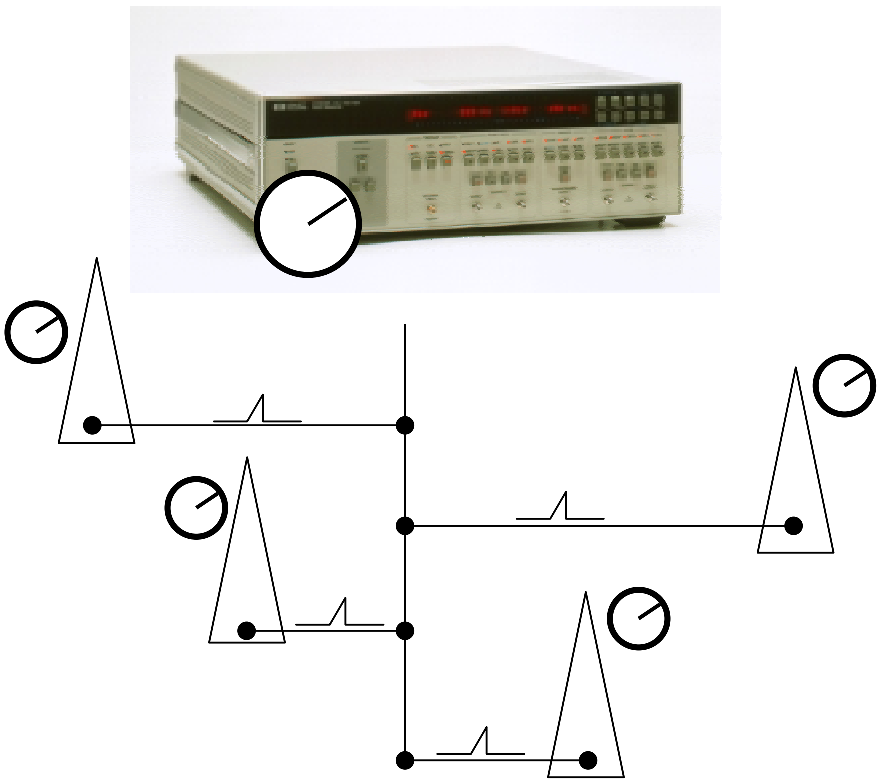


Figure 10: Common time base solution

The common time base shall be considered a pulse generator, producing GHz-pulses at a very high accuracy. That is intentionally a vague formulation, since the ultimate goal *always* is to reduce the errors at every single source of errors. The clocking is *one* among many errors (multipathing, fading, deviations in signal interception, finite accuracy at the calculating servers, etc.) and what ultimately decides the accuracy is the cost of such equipment, assuming that the technology providing the desired accuracy exists. To meet our needs, the pulse generator should be able to work at a frequency $f \geq 1$ GHz to provide the necessary resolution for our problem.

The diagram below gives a very clear picture of how the corresponding frequency (Y-axis) on our counters exponentially increases as the desired resolution decreases (X-axis).

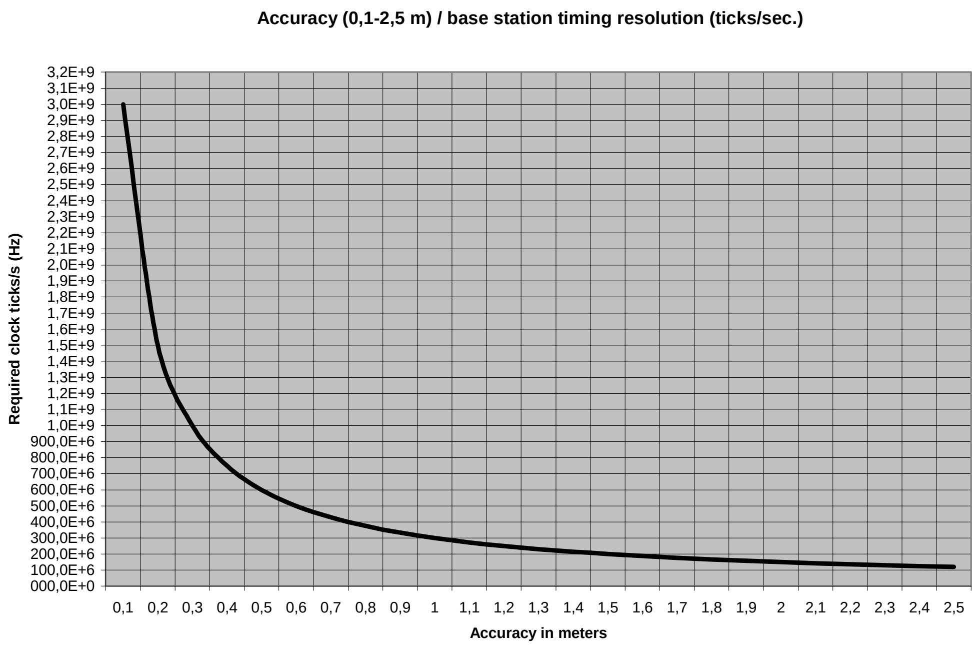


Figure 11: The need for high frequency due to high resolution

As can be seen above, we rapidly move into the GHz-domain below 0.3 meters of resolution, but also how we become less sensitive to errors in frequency (i.e. vertical movement on the y-axis) at high frequencies.

3.2.2.1. *What market provides for a centralized solution*

Agilent Technologies for instance, provides equipment with the required resolution described in section 1.2.1 by means of the Agilent 8133A Timing Generator[8], at $37080 (29/03/2000).

Figure 12: The Agilent 8133A Timing Generator

This is, both in terms of functionality, price and accuracy, an overkill, but it proves that the market provides even better equipment than we actually need. It also offers various output frequencies, friendly user interface, etc; features which are not necessary for our application.

```
Timing
Frequency: 33.0 MHz to 3.0000 Ghz, 100 KHz resolution
Period: 300 ps to 30.000 ns, 1 ps resolution
Accuracy: ±0.5%, ±0.1% nominal
Internal/External clock
Continuous running (no trigger modes)
```

Figure 13: Extract from the Agilent 8133A specification

A pulse generator like the Agilent 8133A is typically used for benchmarking and testing, and during a test-stage it would definitely be suitable to use such a device both because of its flexibility in terms of various output frequencies and also because of its extreme accuracy.

[8] See http://www.tm.agilent.com/tmo/datasheets/English/HP8133A.html

The counters at each base station also have to meet the GHz-demand and one company dealing with such high-performance technology is Japanese NEL[9]. Their NL4609-2[10] is a three-stage ripple counter consisting of three serially connected GaAs technology based flip-flops. To get a feel for the price level of the NL4609-2 as of today (April 2000), according to [25] they start at $1024 each (5-9 pieces) and $829 (100-999 pieces). Price exponentially declines at higher volumes.

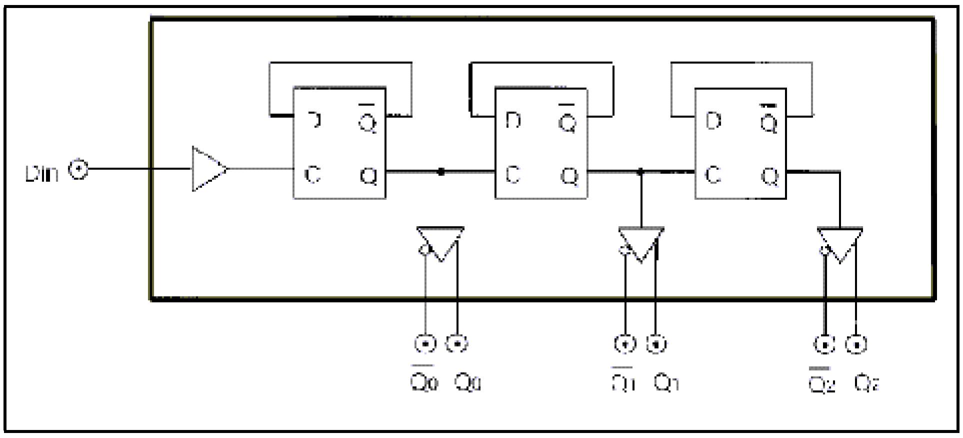


Figure 14: The NL4609-2 three-stage Ripple Counter

By providing a 32-bit counter, we could perform measurements during one second on a 2 GHz signal. By serially connecting 11 three-stage counters like the NL4609-2 ripple counter we achieve a modulo-34 bit counter. In such a cascade coupling, the input signal would be the least significant bit, Q0 the second last significant bit, etc. Also, by observing that the frequency of the output signal Q is halved at each flip-flop, we do not have to use high performance circuitry throughout the cascade, but rather, a clever mix of cheap components for the low-frequency regions near the most significant bit and high-performance components (like the NL4609-2) closest to the least significant bit.

[9] See http://www.nel.co.jp/english/index.html
[10] Data sheet etc, see http://www.nel.co.jp/uhe/ic_data/nl4609-2.html

3.2.3. Conclusion

The synchronization issue is very crucial to any TDOA system, but the comparatively hard constraints in this application, makes it a real challenge. There are distributed algorithms available, implemented with or without time dedicated hardware, but none of them can meet our demands. Mills states [18] that “A time resolution of 10 ns with a 100-Mb Ethernet is in principle possible, but probably not in practice without special hardware”. Likewise, the HP patent can not meet the demand either.

The only thing that can be concluded after this discussion is that we are entering a new, almost unexplored “nanosecond-territory” where much more has to be done to meet our demands. The problem seems to be very hardware dependent and hopefully the future can give us hardware with necessary resolution accuracy.

As mentioned before, we will subsequently assume a synchronized network with a timing accuracy < 8 ns.

3.3. TDOA in depth

TDOA relies on so-called *hyperbolic position location,* which is performed in two steps. First of all, the actual time differences between pairs of participating base stations must be estimated. Second, these time differences are being transformed into a set of nonlinear hyperbolic equations, which ultimately yields the position of the actual PDA. We start off with the TDOA estimation.

3.3.1. Step 1: Estimating the TDOA

Three ways of estimating the TDOA will be considered:

1. One *Time-based TDOA on direct triggering* (page 30) approach is to stop (or read out, i.e. not necessarily stop) the clock at each base station as soon as the signal reaches it (by defining a certain power-level), and decide which PDA sent it. This can be done in a couple of slightly different ways. Assuming we manage to stop the clocks accurately, we send (over Ethernet or whatever communication infrastructure is provided) the *clock's time* to a central unit, which calculates the position by collecting the times from the other base stations together with the known positions of the base stations.

2. The second method, *TDOA with distributed, continuous cross-correlation* (page 40), is to use the same bit-stream as in 1, but we stop the clock *after* the cross-correlation has been performed and the correct signal has been identified. This scheme requires additional hardware for the cross-correlation, which preferably is done in an analog manner (if digitally, additional high-frequency sampling hardware must be provided).

3. The third approach is the *TDOA by centralized cross-correlation* (page 41), which relates to *TDOA with distributed, continuous cross-correlation* in 2, but the centralized solution will have another impact on the system design - in terms of high-frequency sampling (if it is done digitally, which, in turn, would ease the further processing of the data). Furthermore, a more detailed, mathematical treatment of the cross-correlation technique is presented in this section (3.3.1.3).

3.3.1.1. Time-based TDOA on direct triggering

There are 79 available frequencies (except for France and Spain) available and hops are made 1,600 times/second according to the Bluetooth specification [6].

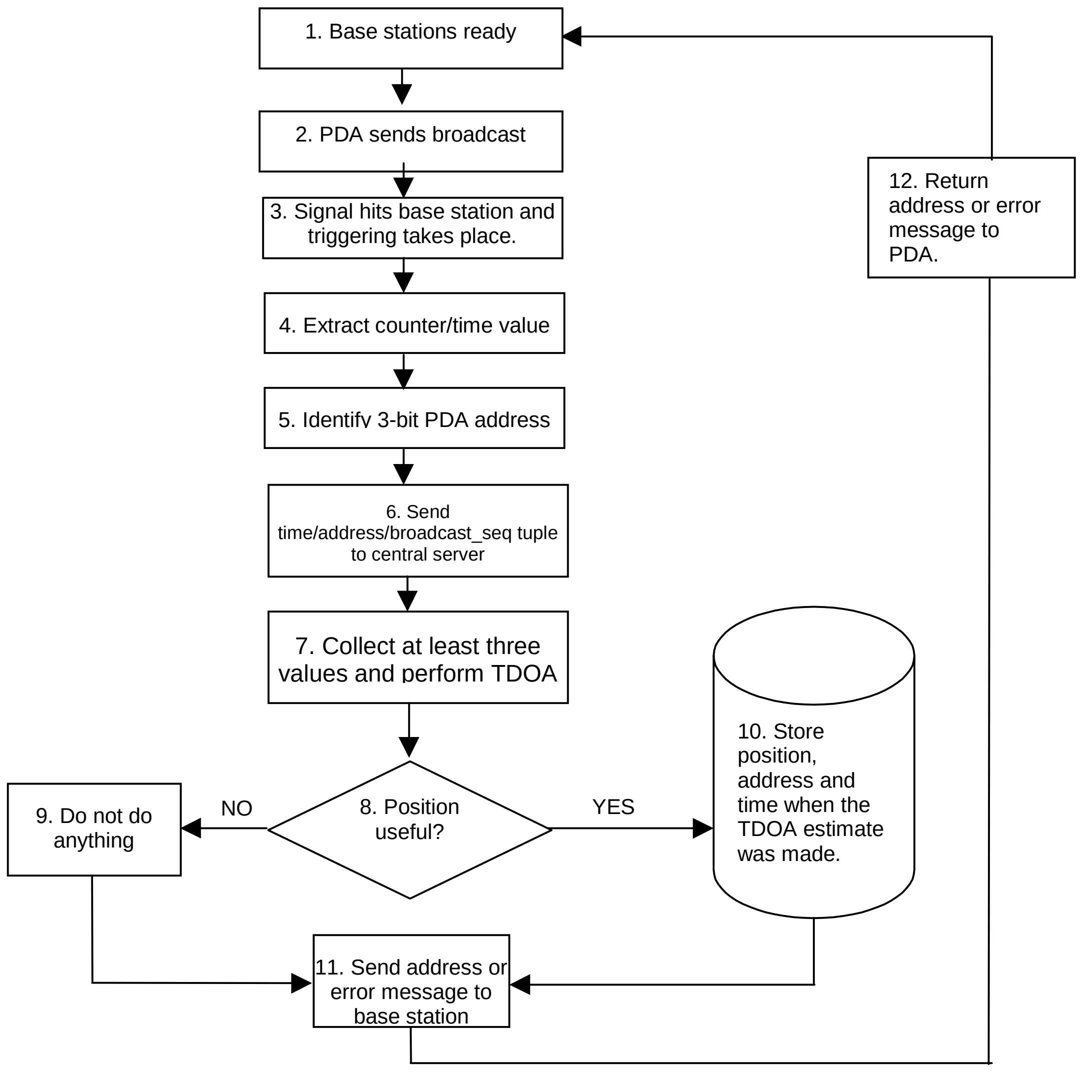

1. Base stations ready
2. PDA sends broadcast
3. Signal hits base station and triggering takes place.
4. Extract counter/time value
5. Identify 3-bit PDA address
6. Send time/address/broadcast_seq tuple to central server
7. Collect at least three values and perform TDOA
8. Position useful?
NO
YES
9. Do not do anything
10. Store position, address and time when the TDOA estimate was made.
11. Send address or error message to base station
12. Return address or error message to PDA.

Figure 15: Event diagram for time-based solution

This first, as it may seem somewhat naive approach, does have a number of appealing features: if we assume that the participating PDAs frequently (let's say at least every 0.1 second) sends labeled messages (i.e. messages containing a valid address that uniquely identifies the PDA) that can be intercepted by at least three base stations, we do not have to generate any extra control traffic to achieve our positioning. The idea behind this scheme is to stress the importance of the accurate timing by "moving" the time triggering as close to the antenna as possible [20], avoiding jitter caused by sampling or even by the operating system. This scheme requires that we trigger on *any* incoming signal and these signals may well be signals not of interest. Filtering and demodulation will take away most of the noise and we can be confident that the signal, whether or not it's of interest for the sake of our positioning, is a valid Bluetooth signal.

A solution like this soon turns out to be very protocol as well as hardware dependent. "Any" signal is a vague definition, which is not enough to make an exact system specification. This "any" must, by some means, be supported and implemented in the protocol used by the system. Therefore, we will subsequently assume a pure Bluetooth environment.

For a typical Bluetooth chip, the trigger hardware is preferably probing the incoming signal after demodulation and filtering, but *before* the digital sampling [26]. The sampling in Bluetooth is typically done at 1 MHz, which would drastically reduce the resolution and discard valuable information of the original signal.

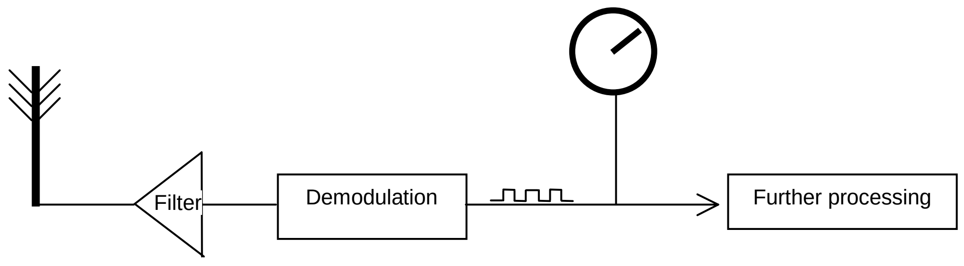


Figure 16: Time-based system overview

By doing this, most of the noise has been filtered out and we have a pretty "clean" signal to work with (not as good as Figure 16 suggests though - it is just a sketch). Of course, compensation for the delays caused by filtering and demodulation must be taken into account. According to [24] their mean delay time is well defined and the deviations are very small (compared to our needs i.e. of the order of picoseconds).

It should be stressed that such small, fixed errors can be compensated for by calibration. What hurts us the most, are large, rapidly varying deviations.

The second advantage is the fact that we can use the standard bit-stream for identifying the PDA. This can be done at application-level, since the critical stop-time already was captured.

This scheme could, in principle, be applied to any technology supporting some radio based protocol that allows broadcasting, means of uniquely identifying the units and some technique (like for instance frequency-hopping) which prevents us from capturing signals not of interest.

In Figure 15, a suggested event diagram sketches this scheme, and comments will now follow to further describe each stage. The exact Bluetooth and timing considerations are described in *A proposed system design* on page 59. Numbers below refer to the numbers in Figure 15.

1. "Base stations ready" means that at least three base stations are ready to "listen" to and trigger an incoming signal. It is important to provide mechanisms that do NOT trigger on outgoing signals. This could be controlled by the baseband unit in the Bluetooth case.

2. PDA sends a signal. For this signal to be of interest, it must be a signal capable of being intercepted by several (at least three) base stations. This could, in Bluetooth, be solved by letting (forcing) the PDA to become a master. This is possible by doing a so-called Master-To-Slave switch. In addition, it is possible for the PDA to be both a slave and a master: we can think of the PDA (master) as forming one piconet with the base stations (slaves). Still, the PDA can be part of another piconet as a slave. This overlapping net structure is called a *scatternet* [6].

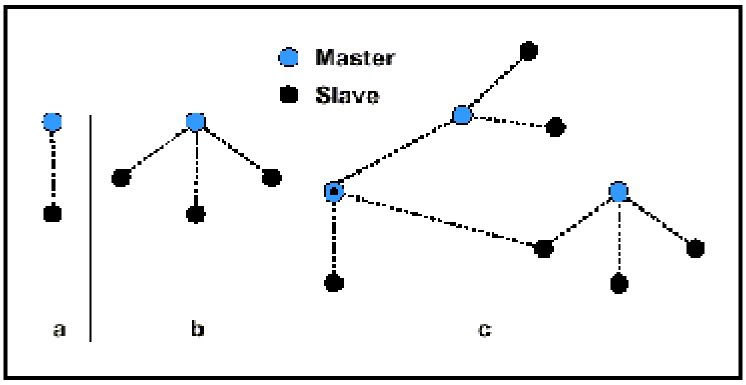


Figure 17: a) Single slave operation, b) multi-slave operation and c) a scatternet operation (from [6])

This scheme will need a software plug-in for the PDA that forces the PDA to become a master. Secondly, we want the PDA to perform a broadcast as soon as the PDA wants to do a position inquiry. Both these actions can be performed programmatically at the application level.

The broadcast performed by the Bluetooth master is not acknowledged by the receiving units. To increase the probability of reaching all reachable units, the broadcast is sent repeatedly according to:

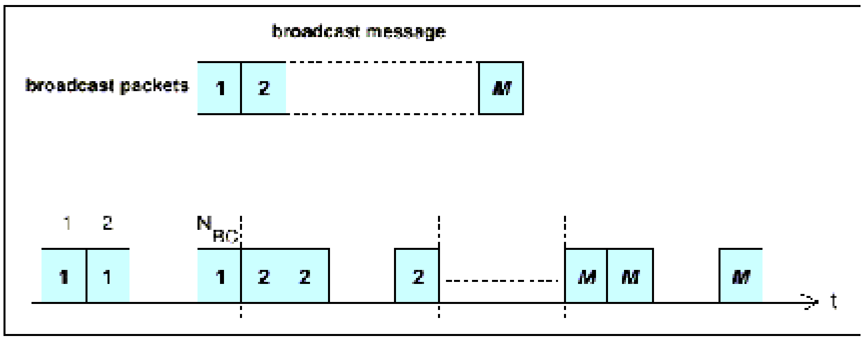


Figure 18: Broadcast repetition scheme (from [6])

As depicted in Figure 18, the Bluetooth specification suggests a set of messages all numbered from 1 to M. These are, during the broadcast phase, transmitted repeatedly.

3. The signal hits the base station and triggering takes place. The most critical time point in this scheme. A rule of thumb common to most high-performance synchronization hardware is to keep it as close to the medium as possible [27] to avoid further introduction of jitter. The threshold level must be set equally on each base station, and its value must be decided empirically depending on the characteristics of the filter and signal strength of the unwanted noise. Since we are probing the signal *after* demodulation and filtering, we can hope[11] for the incoming signal to be a signal of interest. Since the PDA is the master, it is controlling the traffic in each piconet it belongs to, respectively, and can decide when to send and not to send.

[11] Since we can never know what the environment looks like in a real situation, there is no guarantee that the incoming signal is what we want. Interference will most likely occur and retransmission will be required.

4. The counter may as well be stopped instead of doing a "snapshot" of the current value at the triggering. The value is typically stored in a register for later processing and transmission to the main server which does the TDOA estimate. Counter hardware capable of working at GHz speed was introduced on page 27.

5. A very appealing feature of this solution is the fact that the ordinary Bluetooth protocol can be used for identifying the PDA. This can be done at the application level with as high reliability as the protocol provides. No extra hardware is needed for the identification in the mobiles.

6. The address of the PDA (the master) together with the time value *and* the Broadcast sequence number is sent from the base station to the server over the media of choice for the infrastructure, typically Ethernet. Every broadcast message has its own sequence number and this will help us to match the correct messages at the server side. Since the piconet only can contain one master and the address of that master (the actual PDA) is known, it is not likely to be a problem to find out which PDA is doing the position request. For a more detailed description on how this identification and addressing is solved, see *A proposed system design* on page 59.

7. At the server, a matching of the same PDA addresses and broadcast sequence numbers takes place and the position is being calculated. The position together with information *when* the calculation is being performed (according to the local clock of the server - that accuracy will suffice since we relate it to the speed of the positioned object, not to the speed of the signal) is stored.

8. To decide whether the position is useful or not can be done in several ways. First of all, if the position is out of the physical coordinate system defining the complete environment, the position can, for obvious reasons be discarded immediately. Second, *if* (which is not evident) we have information about the previously calculated position, we can, provided the new position, calculate the average speed of the user/object. If this speed exceeds a certain value, we can assume the position is wrong (if the speed is unrealistic for the particular object). To further refine this latter method, if knowledge of the type of object is known (fast moving objects on a conveyor belt, pedestrians, etc), different threshold speed values can be applied to different objects. This way of discarding certain position will also enable us to use less than three base stations. For instance, if two base stations detect a PDA in an aisle, that will be sufficient for us to calculate the position.

9. If the position is not useful, we basically do nothing in terms of storing or further processing of the data.

10. If the position is found to be realistic and useful in a sense defined by the criteria discussed in 8, the position (coordinates) together with the Bluetooth address (BD_ADDR) is stored in a simple, but preferably fast access database.

11. If the address is to our satisfaction, we return an address or else send an error message to the base station.

12. Finally, we return either an error message or position to the requesting PDA. The PDA may do a re-request of the position if something failed.

A very important observation with scheme is that, since we are probing the incoming signal as close to the medium as we are, not only the signal being broadcasted from the master PDA will be detected by the base stations, but also all other PDAs transmitting in the same area *and* on the same frequency, regardless if they are “logical” Bluetooth masters or not. This interference is basically caused by the probability that these unwanted PDAs happen to transmit at the same frequency (decided by the frequency hopping) at the same time due to the random behavior of the hopping-frequency pattern. Now, one may be tempted to conclude that we do not need the Master-To-Slave switch recently described, but in order to increase the probability that we will be able to establish a common hopping-frequency pattern and by doing that, reach the base stations, we have to somehow agree upon a certain hopping frequency pattern. This is all done at the time of the establishment of the piconet.

The price we have to pay for the simplicity of this scheme, is that the behavior of the system will be statistically determined (and therefore unpredictable to some degree), depending on the amount of PDAs which are active in the area. No doubt, there will be a correlation between the amount of PDAs (regardless if they act as masters or not) in the area and the performance of the system in terms of retransmissions.

To build up and analyze a realistic model of the scenario to its full extent is out of scope for this paper, but let’s stick to this topic a little more since it is of great importance to the implementation of the system. Assume a Bluetooth base station in low-range mode. It covers, at least theoretically, an area of about 314 m^2. Let’s further assume a density of PDAs of one PDA per square meter.

It follows directly that we have 314 active PDAs to deal with *plus* at least three base stations. Now, let's go on with a worst case scenario: all PDAs are active in some piconet. This really is a worst case scenario, since many of them, when not in use, will enter the other states defined in [6]: *park*, *standby*, etc. These various modes drastically reduce the transmitting and the key incentive for introducing such modes is power saving (primarily for the handheld PDA). In addition, it reduces the amount of traffic in the piconet, which in turn lets us use available frequencies more efficiently.

317 units, and each and every an active member of some piconet, yields 46 piconets. One of them is our piconet of interest for the positioning, in which the base stations participate. This implies that we have 45 other nets, or hopping-frequency patterns if we like, that may interfere with our piconet of interest. Assume also, that the probability for each and every of these nets choosing the same frequency as "our" net has chosen is $1/\text{amount\_of\_available\_freq}$. For the standard Bluetooth net (except for Spain and France), that means a probability of $1/79 = 0.01266 \approx 1.3\%$. 45 nets result in a total probability of $45/79 = 0.56962 \approx 57.0\%$ that we will detect a collision at any particular instant. Or from the other point of view: we have a 43.0% chance to capture the correct signal! Since we are repeating the broadcast transmission several times (the exact number of times is *not* specified in [6], nor is the maximum number of messages) we drastically increase the chance of getting the correct signal.

> ”Since broadcast messages are not acknowledged, each broadcast packet is repeated for a fixed number of times. A broadcast packet is repeated N_{BC} times before the next broadcast packet of the same broadcast message is repeated.”

Figure 19: Quote from [6]

The quote made in Figure 19 states that each message is sent N_{BC} times until the next broadcast message is sent, and the number of broadcast messages is M. That means that we are repeating our broadcast $M \cdot N_{BC}$ times.

The above reasoning is just *one* way of approaching the problem. Depending on the model we choose (user behavior, number of PDAs per square meter, etc.) we will get new results each time.

A total different approach to the solution described here, would be to process *all* incoming signals, to “hope” for at least three base stations to receive the signal, to decode and adjust who sent it, and finally to calculate the position. This is more of a brute-force solution, which gives us even less control over transmissions and finally performance, but it would not require any piconet creation, Master-To-Slave switch, etc.

3.3.1.2. TDOA with distributed, continuous cross-correlation

Another, slightly different technique, is to store the expected master's address (i.e. bit-pattern) at the base station and perform a cross-correlation at every base station. This solution, though, requires additional hardware in terms of high-performance cross-correlation at *each* base station in which the incoming signal is compared to those in a register of bit-patterns. When the cross-correlation function peaks, the clock corresponding to the actual PDA stops.

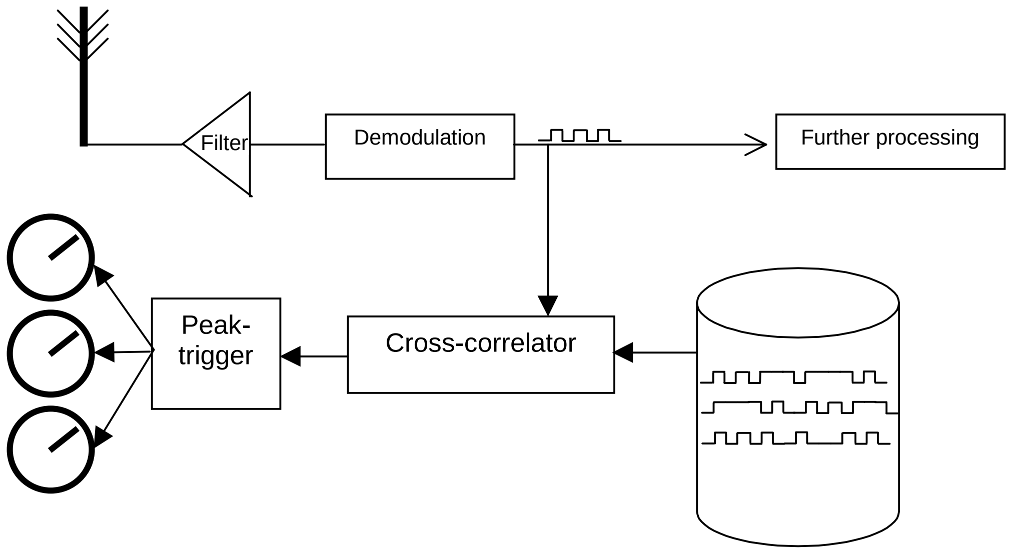


Figure 20: System overview for a distributed, cross-correlation solution

This scheme is appealing if we expect a burst of unwanted traffic in a noisy environment and want to track down a specific sender. However, this is not likely to be a problem with a proper configuration of the logical piconet, letting the PDA be the master and the base station the slave. Increased complexity typically introduces a higher probability for errors.

The essential difference with this scheme is that we do not benefit from the Bluetooth protocol since, in principle, timing and identification is done in one step at hardware level by means of the cross-correlation.

The cross-correlation is typically done in a maximum-likelihood manner and the peak-trigger in Figure 20 adjusts what degree of signal pair resemblance should stop the clock and also which clock. The fact that we do not have to use Bluetooth makes this scheme more powerful, since it generalizes to any other protocol.

3.3.1.3. TDOA by centralized cross-correlation

Probably the most common[12] way to perform the time difference estimate is by means of cross-correlation in which each and every signal pattern from every base station is compared to a referential base station (the base station where the signal first hits the infrastructure). Cross-correlation can be intuitively understood by a simple sketch.

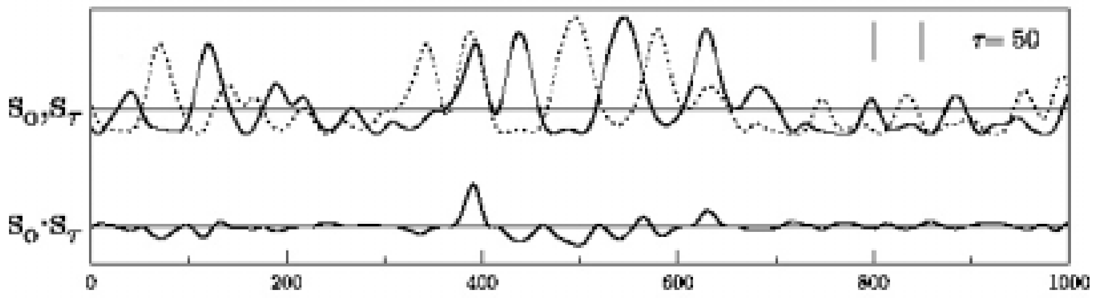


Figure 21: Cross-correlation

The top section shows two identical signals which are phase shifted in time by $\tau=50$ time units. The bottom section shows the result of simply multiplying these two signals. A peak section near 400 can be detected. A special a case of cross-correlation is auto-correlation in which a copy of a signal is compared to itself. It may be tempting to use auto-correlation in our context since our signal originates from the very same source, but since the signals are most likely to be distorted, interfered by other signals (in a noisy environment), multipathed, etc (especially in an indoor environment), we will use the cross-correlation approach.

3.3.1.4. *Cross-Correlation - a brief mathematical discussion*

Even though this paper primarily focuses on a general solution for solving the position problem, error analysis, a system design proposal and not so much about signal specific issues, something ought to be said about the mathematical treatment of the cross-correlation technique. Several studies [28, 29, 30, 31, 32] that treats this subject related to the TDOA problem can be found, and among these, the work of W. A. Gardner should be stressed.

Assuming a disruptive environment, a signal s(t) being emitted from a PDA, a general model for the time-delay estimation between signals at base station x(t) and y(t) will be:

$$
\begin{aligned}
x(t) &= A_x s(t - d_x) + n_x(t) \\
y(t) &= A_y s(t - d_y) + n_y(t)
\end{aligned}
$$

Eq. 5

in which A_x and A_y corresponds to the amplitude of the signals at base station x and y respectively and $n_x(t)$, $n_y(t)$ to noise. To simplify the equation somewhat, we can use x as a reference station, assuming that the signal hits x first (i.e. $d_x<d_y$):

$$
\begin{aligned}
x(t) &= s(t) + n_x(t) \\
y(t) &= As(t - D) + n_y(t)
\end{aligned}
$$

Eq. 6

where A equals A_y/A_x and $D = d_y - d_x$ where ($d_x<d_y$). The amplitude A may also give us a hint of which base stations to select. Also assumed in this simplified model is that $s(t)$, $n_x(t)$ and $n_y(t)$ have zero-mean (time average) values and that $s(t)$ is statistically independent (over time) of $n_x(t)$ and $n_y(t)$.

[12] The author's own experience and also the opinion of [36].

It follows [31] that the limit cyclic cross correlation and autocorrelations are given by (complex notation):

$$
\begin{aligned}
&R_{yx}^{\alpha}(\tau) = AR_{s}^{\alpha}(\tau - D)e^{-i\pi\alpha D} + R_{n_x n_y}^{\alpha}(\tau)\\
&R_{x}^{\alpha}(\tau) = R_{s}^{\alpha}(\tau) + R_{n_x}^{\alpha}(\tau)\\
&R_{y}^{\alpha}(\tau) = |A|^2 R_{s}^{\alpha}(\tau)e^{-i2\pi\alpha D} + R_{n_y}^{\alpha}(\tau)
\end{aligned}
$$

Eq. 7

α is called the cycle frequency and it describes the periodicity of the correlated signals.

The previously described mathematical description of the problem is useful when approaching the different techniques used for estimating the variable of interest: D. Cyclostationary techniques surpass the Generalized Cross-Correlation (GCC) *only* when the present noise and interference exhibit a cycle frequency different from the signal of interest [33]. Among the cyclostationary methods we find the Cyclic Cross-Correlation (CYCCOR), the Spectral-Coherence Alignment (SPECCOA) method, the Band-Limited Spectral Correlation Ratio (BL-SPECCORR).

For sake of clarification: a cyclostationary process is one which has statistics which vary periodically with time: the mean, autocorrelation, 3rd, 4th ... order cumulants are all periodic function of time. An example of a first-order cyclostationary process is a sinusoid with added Gaussian noise: the noise clearly causes the signal to be random. However, if the signal is averaged by matching up points from the same position in the cycle each time (synchronous averaging) then the underlying sinusoid will be recovered. If the variance of the signal is computed in the same way, it will be found to be constant.

3.3.1.5. General Cross-Correlation system scenario overview

Before any cross-correlation can occur, ordinary preprocessing [34] such as filtering has to take place. This is done in order to localize frequencies with high Signal-To-Noise Ratio (SNR) and to eliminate (if possible) noise or other unwanted signals. Next, the two signals are multiplied, integrated, squared and finally the peak detector returns our estimate of D, denoted $\underline{D}$, after performing a set of time shifts (τ). Remember: the cross-correlation functions are *not* functions of time, but rather of the τ-delays! The filter functions are denoted H(f) below.

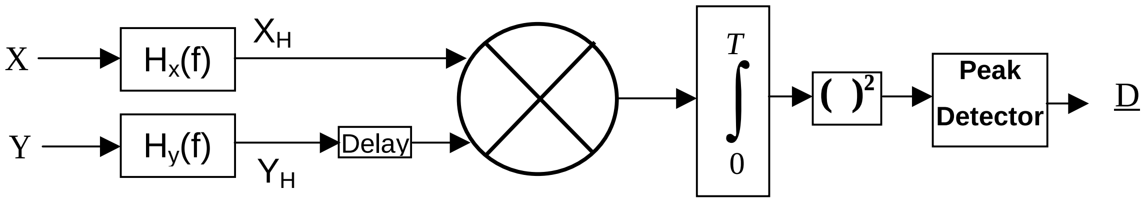


Figure 22: TDOA Generalized Cross-Correlation (GCC)

The generalized Cross-Correlation assumes that the cycle frequency α=0 [31], and we can re-relate to Figure 22 by defining the Cross-Correlation relation between *x(t)* and *y(t)* which returns D, simply by finding the argument τ that maximizes the function:

$$R_{y,x}(\tau) = R^{0}_{y,x}(\tau) = \int_{-\infty}^{\infty} x(t)y(t-\tau)dt$$

Eq. 8

In practice though, $R_{y,x}(\tau)$ can only be estimated during a finite time interval:

$$\underline{R_{y,x}}(\tau) = \frac{1}{T}\int_{0}^{T} x(t)y(t-\tau)dt$$

Eq. 9

Filtering in communication systems typically rely on the analysis of *power spectral density* (PSD), giving us the possibility to calculate the distribution of the power of our signal in the frequency domain. There is also a close connection between PSD (or rather: *cross* power spectral density function) and cross-correlation:

$$R_{x,y}(\tau) = \int_{-\infty}^{\infty} G_{x,y}(f) e^{j\pi f\tau} df$$

Eq. 10

In other words: they are Fourier transforms of each other and the corresponding integral that takes us from the cross-correlation function to the PSD (from the time domain to the frequency domain) is:

$$G_{x,y}(\tau) = \int_{-\infty}^{\infty} R_{x,y}(f) e^{-j\pi f\tau} dt$$

Eq. 11

As mentioned earlier, the signals x(t) and y(t) are filtered by using a filter function:

$$\Psi_G(f) = H_x(f) H_y^*(f)$$

Eq. 12

By using the formulas we have so far, we can finally create the estimated, pre-filtered cross-correlation function which in turn gives us the TDOA D:

$$\underline{R^G_{y_H,x_H}}(\tau) = \int_{-T}^{T} \Psi_G(f) \underline{G_{y,x}}(f) = e^{j\pi f\tau} df$$

Eq. 13

Underlined items denote estimations (since we can only integrate over finite intervals). One of the crucial things for successful TDOA estimation is the choice of the filter function $\psi_G(f)$.

The objective is always a nice, well defined peak to help us estimate the TDOA D, but a large peak results in a narrower spectra which in turn is subject to error introduction. As [35] states, a trade-off must be done between resolution and robustness against errors, when choosing $\psi_G(f)$. The same author also provides examples of different functions: their drawbacks and advantages respectively.

3.3.1.6. lesswire considerations and conclusions considering a cross-correlation scheme

The logistical problems of the cross-correlation for lesswire's infrastructure are related to the fact that the cross-correlation must occur in some dedicated central unit, or at least "at the same place" in some sense. This signal collecting takes time, bandwidth, and computer resources. During this transfer process, a probability of further introducing errors is possible, which destroys the signal data and finally our TDOA D estimate.

The problem of receiving all signals - also signals not of interest - is likely not be a major problem as mentioned before, due to Bluetooth hopping-frequency scheme (1600 hops/s.) which drastically reduces the probability of interference.

3.3.2. Conclusion

For the lesswire position system to be fully implemented and reliable, a set of three base stations must cover those spots where position requests are to be calculated. Considering the Bluetooth standard range of 10 meters, the amount of base stations tends to be large. There is a possibility of increasing [6] the output power of the Bluetooth devices. This can be easily done at the base station side, since power consumption is not likely to be a problem, but we can not assume that the PDA supports this. Neither can we force the PDA to switch to high-power mode. The PDA will, in most cases be battery powered and hence, the standard 10-meter range will be used. To cover a 50 000 m^2 area, approximately 1 000 units would be needed[13]. Hence, there is a very strong incentive to keep the unit costs as low as possible.

The cross-correlation techniques all introduce more complexity and costs. The costs for the distributed cross-correlation solution are directly proportional to the amount of base stations. As concluded on page 27, GHz-counters are for the time being comparatively expensive components. A slight modification of this scheme only uses one clock and only one register with the known address of the PDA. The difference between this scheme and the *Time-based TDOA on direct triggering* in terms of extra equipment is the auto-correlation mechanism and in the ability to treat several signals: more clocks and an intelligence that stop the correct clock.

[13] If the base stations are *optimally* arranged, every new base station increases the coverage area by $r^2\pi/6 \approx 52$ m^2.

For managing the centralized solution, we must provide a means for either high-frequency sampling of the signal (digital cross-correlation) or making a “snap-shot” of the signal for further processing (analog cross-correlation). The digital solution gives us the possibility to easy manage the processing and transmission to the central server where the cross-correlation takes place.

To implement a position system that is economically feasible, comparatively easy to integrate with Bluetooth software, as well as hardware, we conclude that the *Time-based TDOA on direct triggering* solution is the most realistic solution.

3.3.3. Step 2: Solving the hyperbolic equations

We are now moving from step one of finding the TDOA estimates - a problem which focuses on signals, timing and hardware - to a purely algebraic treatment and our core problem: calculating the position of the PDA. Given, from step one, the TDOA estimates (according to the proposed *Time-based TDOA on direct triggering*) we can define a set of non-linear hyperbolic equations.

3.3.3.1. *A geometrical and mathematical model - understanding TDOA*

Let's return to the scenario in section 2.1.1.3, but this time with a more complete mathematical treatment. The black circles model how the signal receiving by each base station can be interpreted: a black circle around a base station implies that somewhere on the circle's perimeter, signals from an active PDA have been detected.

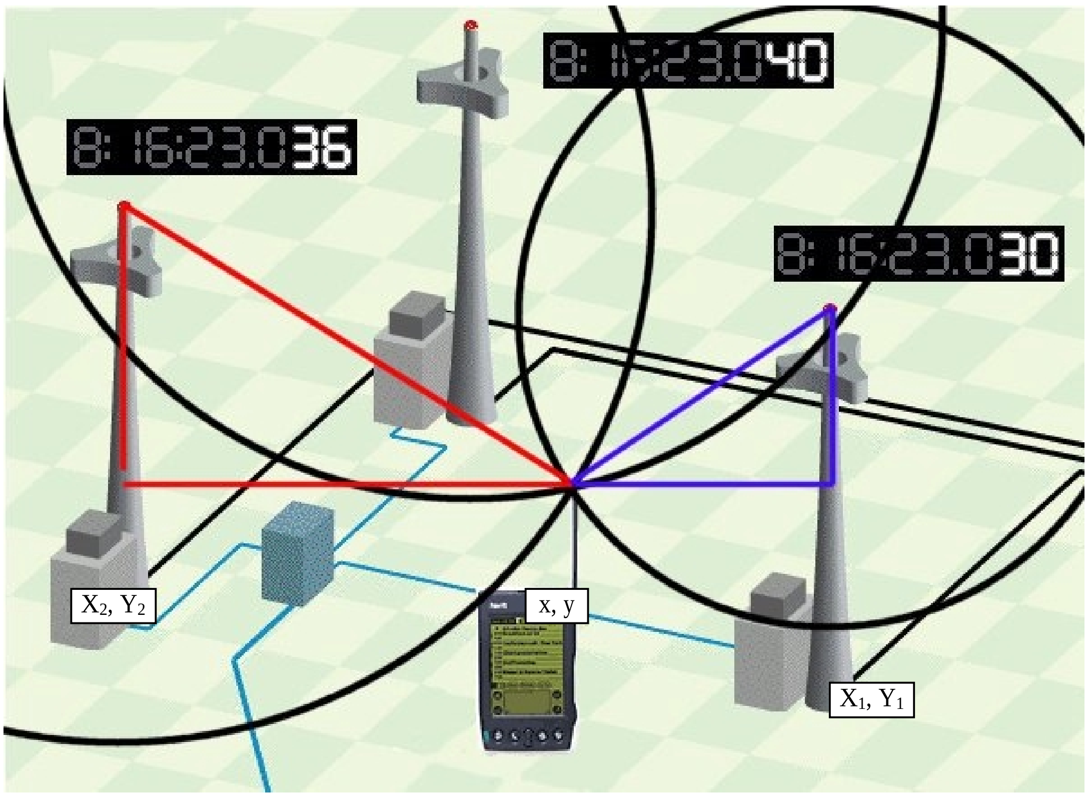


Figure 23: TDOA and geometrical model

It is obvious from studying the figure[14] that three base stations are needed to achieve a unique solution for 2D. Only two base stations (circles) give us an undesired two-point ambiguity. The model can easily be expanded to 3D, but that will take four base stations. Furthermore, although three base stations are available, they must not be arranged in a straight line, since we will still have an ambiguity. This special case is not treated in this paper.

Let (x, y) be the unknown position of the PDA, X_i, Y_i the known position of base station i= 2…N (we treat base station number 1 as the reference station). From the labels in Figure 23, we can easily define the squared range difference between the PDA and the *i*th receiver:

$$R_i = \sqrt{(X_i - x)^2 + (Y_i - x)^2}$$

Eq. 14

Now, the range differences between the base stations and the PDA equals the TDOA D, and relating this to the reference base station, we get:

$$R_{i,1} = cd_{i,1} = R_i - R_1 = \sqrt{(X_i - x)^2 + (Y_i - x)^2} - \sqrt{(X_1 - x)^2 + (Y_1 - x)^2}$$

Eq. 15

Where c is the speed of the signal and $d_{i,1}$ the TDOA estimate. These equations together form the set of nonlinear equations that finally yield our position estimate of the two unknown parameters x and y.

It turns out to be a difficult thing to solve these equations and a common way to solve them is by first linearizing them. In this paper, a short overview of different techniques will be investigated, and two alternative, interesting approaches will be examined closer: one, which gives the solution explicitly in closed form and one iterative solution.

14Due to the feeling of depth in the picture, the infrastructure simulates a 3D-scenario, but the geometrical figures of interest assume a pure 2D-scenario.

3.3.3.2. Different methods for solving the set of non-linear equations

Several methods for solving the nonlinear equations have been proposed. We primarily want to satisfy the following two conditions:

1. We want a **fast** algorithm to be able to process requests in a short period of time.
2. We would like to **benefit from several base stations** (more than three in the 2D-case and more than four in the 3D-case).

In this paper, we will focus on two different methods. These are Chan's method, which has attracted a lot of attention lately, for instance in [27], and a maximum-likelihood approach, which works iteratively using the Levenberg-Marquardt minimization [36].

The Taylor-Series method linearizes the set of equations by Taylor-Series expansion and then iteratively solves the system [37]. The method requires an initial guess and improves the estimate at each iteration by determing the local linear least-square solution. It also benefits from redundant measurements. The drawbacks are the computational effort and the initial guess must be good. Another drawback with linearization is that it *can* introduce significant errors under bad GDOP (Geometric Dilution Of Precision) circumstances. GDOP refers to the situation in which a comparatively small ranging error results in a large position error due to the fact that the PDA may be situated on a part of the hyperbolas far away from the base station.

Chan's method [27, 38] is a non-iterative solution to the problem. It performs, as we will see in *Simulation and error analysis*, significantly better than the iterative Levenberg-Marquardt solution, even though the latter is widely used because of its efficiency (compared to other iterative solutions).

It is somewhat unique in the sense that it provides an explicit solution, unlike the Taylor-series method, and it can take advantage of redundant measurements. The drawback is that it needs further information to solve the ambiguity that arises due to the two solutions generated by the quadratic equation in R1. This ambiguity though, will not cause a problem.

To treat Chan's method mathematically, we assume three base stations producing two TDOA estimates.

$$\begin{bmatrix} x \\ y \end{bmatrix} = -\begin{bmatrix} X_{2,1} & Y_{2,1} \\ X_{3,1} & Y_{3,1} \end{bmatrix}^{-1} \times \left(\begin{bmatrix} R_{2,1} \\ R_{3,1} \end{bmatrix} R_1 + \frac{1}{2} \begin{bmatrix} R_{2,1}^2 - (X_2^2 + Y_2^2 + X_1^2 + Y_1^2) \\ R_{3,1}^2 - (X_3^2 + Y_3^2 + X_1^2 + Y_1^2) \end{bmatrix} \right)$$

Eq. 16

For sake of clarity: $X_{i,j}$ refers to X_i-X_j ($Y_{i,j}$ similarly), $R_{i,j}$ is defined in Eq. 15 and finally X_i, Y_i are the X and Y-coordinates of base station i.
How this solution can be extended to handle more than two base stations is further treated in [38].

We solve Eq. 16 in R1 and substitute x and y in Eq. 14 (by first setting i=1). By solving R1, we substitute back into Eq. 16 whereupon we achieve our final solution for x and y. The ambiguity occurs when solving R1, but it can be shown that one solution always results in positions way out of range. In the error and simulation section, a full simulation of every single combination is made, and no single one suffers from this ambiguity. The root for R1 is always as chosen[15]:

$$R_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Eq. 17

[15] a, b and c come from the standard equation of the second order we get when substituting as described above: $y(x)=ax^2+bx+c$.

The Levenberg-Marquardt algorithm is iterative and requires an initial “good” guess for the iteration to terminate properly and find the solution. Still, though, the returned solution may be a saddle-point, a local minimum, or one or more of the parameter values may tend to infinity. This is of course a drawback, generally speaking, but is most likely *not* to cause any problems to our application, since as soon as the PDA manages to establish connection with at least three base stations, we can make a very good initial guess (given the small-sized Bluetooth picocells).

This algorithm is one of the most widely spread algorithms for solving non-linear problems and it is very efficient in terms of the amount of required iterations. It does not use derivatives of the second order, which makes it less computationally demanding. It uses a trust-region approach, also known as *ridge regression* or *regularization*, to shrink the step sizes to ensure a reduction in our target function at each iteration.

For our problem, we first have to rearrange our equations slightly and “transform” the problem into being a minimization problem:

$$\begin{aligned}
f_1(x,y) &= \sqrt{((X_1-x)^2+(Y_1-y)^2)} - \sqrt{((X_2-x)^2+(Y_2-y)^2))} - d_{1,2}\\
f_2(x,y) &= \sqrt{((X_1-x)^2+(Y_1-y)^2)} - \sqrt{((X_3-x)^2+(Y_3-y)^2))} - d_{1,3}\\
f_3(x,y) &= \sqrt{((X_2-x)^2+(Y_2-y)^2)} - \sqrt{((X_3-x)^2+(Y_3-y)^2))} - d_{2,3}
\end{aligned}$$

Eq. 18

Where $d_{i,j}$ is the measured time difference between base station i and j. Under ideal circumstances, this set of functions yield 0. The Levenberg-Marquardt algorithm takes us as close to zero as possible, given the prerequisites and values of the parameters (i.e. positions of the base stations and time differences).

By studying the equation(s) we can conclude that the time difference (where we are most likely to have our most prominent source of errors) $d_{i,j}$ affects the position linearly, which of course is to our advantage. $d_{i,j}$ in turn, is linearly dependent on the times captured at the base stations:

$$d_{i,j} = time\_BS\_i - time\_BS\_j$$

Eq. 19

Due to the fact that the positions of the base stations are known to an almost arbitrary accuracy[16] (depending on how we measure their positions), they will not introduce errors of great significance. The errors they may introduce, due to erroneous measurement of the positions of the base stations, are fixed and will be eliminated by calibration.

3.3.4. Conclusion

The general Taylor-series method suffers from increased complexity and the GDOP problem and we have chosen the two other candidates for solving the hyperbolic position problem: Levenberg-Marquardt and Chan's method. They were chosen because they are representative of the many solutions proposed by different researchers: one iterative and one providing an explicit solution. To evaluate which one of these is the most appropriate one to satisfy our needs, simulations were used to analyze their behavior. These simulations will be shown in the next section.

[16] Nowadays, equipment letting us measure the position down to the quarter of an optical wavelength are available, but this is where manageability and economy comes in: it must not take too long to measure the positions and it must not be too expensive. Calibration will cancel these fixed errors.

4. Simulation and error analysis

This section is devoted to simulation of our system, assuming the algorithms chosen in previous sections: an iterative approach using Levenbergs-Marquardt algorithm for solving non-linear equations and Chan's explicit method.

One assumption will be used throughout this section: We are working with three base stations at positions (0,0), (10,0), (5,8.7). The somewhat odd y-coordinate for the third base station comes from the optimal placement of the base stations described in *Base station setup* on page 61. The +/- 2 meter constraint is also considered as well as the success ratio of 67%. The simulations were run on a Sun Sparc Ultra-1 with Matlab 5.3.0.10183[17]. Only positions where all base stations where in reach of the PDA and vice versa were considered (subsequently called "valid positions").

4.1. Finding the critical 8 ns

One way of finding the maximum allowed time caused by errors (caused by non-synchronized clocks and/or wrong triggering, assuming the proposed *Time-based TDOA on direct triggering*), is to investigate *all* possible combinations of errors and how they affect the calculated position. The granularity of the steps was 1 ns and we also kept track of the efficiency in terms of time and amount of the floating-point operations (flops) required for the simulation. Since we expect a very dense matrix of base stations, due to the short-range Bluetooth radio, only simulations on three base stations have been investigated. Of course, one suggestion for further work is to evaluate the errors as a function of the number of base stations.

[17] See http://www.mathworks.com

For this first simulation, 4913 valid locations were examined with, as mentioned above, all possible combinations of errors over all base stations. By "tuning" the maximum and minimum errors and running a couple of simulations, we found out that 8 ns is the critical time. This time certainly is critical in many ways: it defines our complete error budget. That translates into; whatever errors may occur (primarily badly synchronized base stations, and bad triggering), they must not exceed 8 ns if we still want to maintain the 67% accuracy. It also sets the requirements on the hardware we need to meet this accuracy.

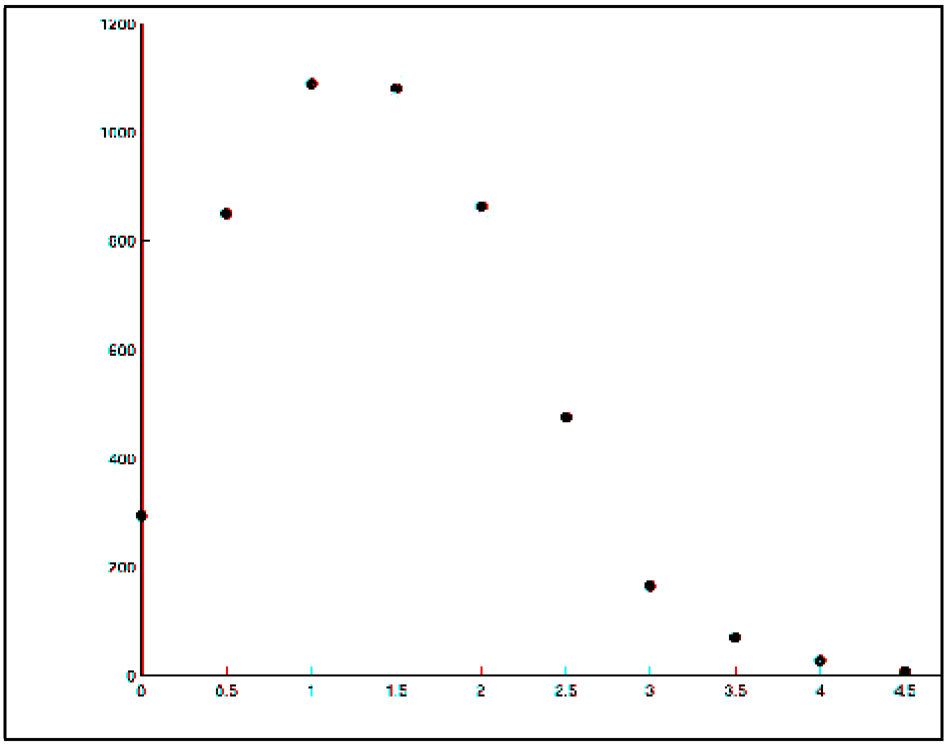


Figure 24: Frequency diagram

The solutions produce the exact same set of results given our prerequisites, but on one point, they differ severely: efficiency. The number of flops needed for Chan's added up to 962 976, whereas Levenberg-Marquardt's needed 15 946 132 – more than 16 times more! The average time/position calculation was 0.0025 seconds for Chan and 0.0472 for Levenberg-Marquardt (almost 19 times slower).

The simulation output and code for both solutions are found in Appendix A.

4.2. Simulating the "real" scenario

The simulation made in *Finding the critical 8 ns* did not give us any feel for how the system will actually behave in the real world. That is to say: let us try to model the "real" scenario in which the PDA enters the infrastructure and wants to know its position.

In this second simulation, we uniformly generate positions for the PDA. All data from the simulation can be found in this section. Under these circumstances, the standard deviations were 1.58 ns (BS1), 4.15 ns (BS2), and 4.27 ns (BS2). Once again, Chan's method outperformed Levenberg-Marquardt in terms of speed and number of required flops.

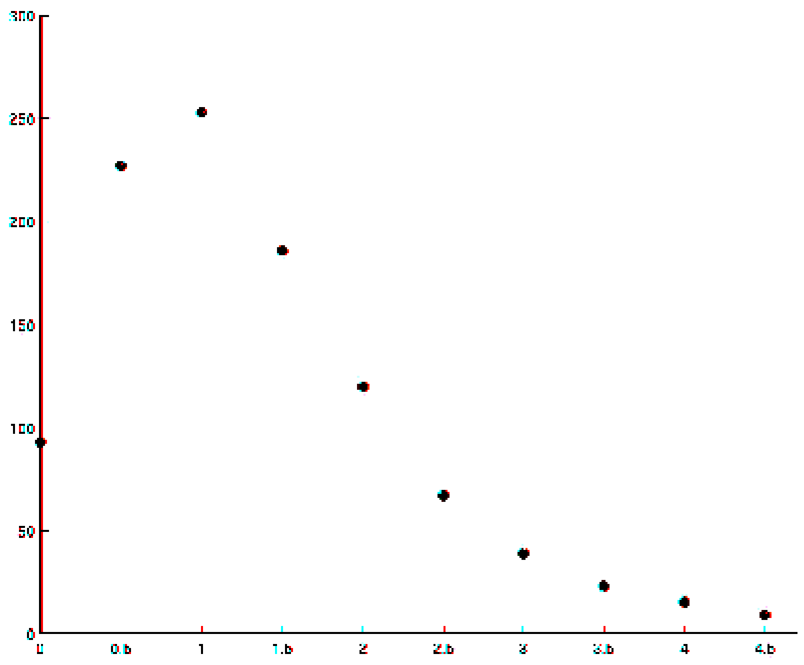


Figure 25: Frequency diagram for Chan and simulation 2

The frequency diagram gives us a pretty clear profile of the behavior. The special class (“error>5 m”) corresponds to 6.4% of the total amount of valid runs and we have a peak in the class (“1 m ≤ error <5 m”).

4.2.1. Conclusion

This section has given us a feel for the tolerance in our system and how it behaves, given our assumptions. The extensive full-search gave us a maximum error/base station of 8 ns. The standard deviations in the second simulation gave us a deviation of approximately 4 ns. Both meet the 67% accuracy. We also can conclude that the choice of algorithm is crucial to the efficiency. Chan outperformed Levenberg-Marquardt in both simulations and is thus the algorithm of our choice.

5. A proposed system design

5.1. Introduction

This section partly serves as a complete summary of the discussions made earlier, and partly more thoroughly describes the complete system design. For sake of clarity, we will consider the lesswire position problem from three different aspects: hardware (overhead circuitry for counters, triggers, and registers), software (Bluetooth considerations at server and client side, timing etc.) and general design considerations (base station arrangement and calibration issues). The synchronization issue is omitted here (see section 3.2 on page 18 for a discussion).

Figure 26 schematically describes the decision process from the original problem to the final solution.

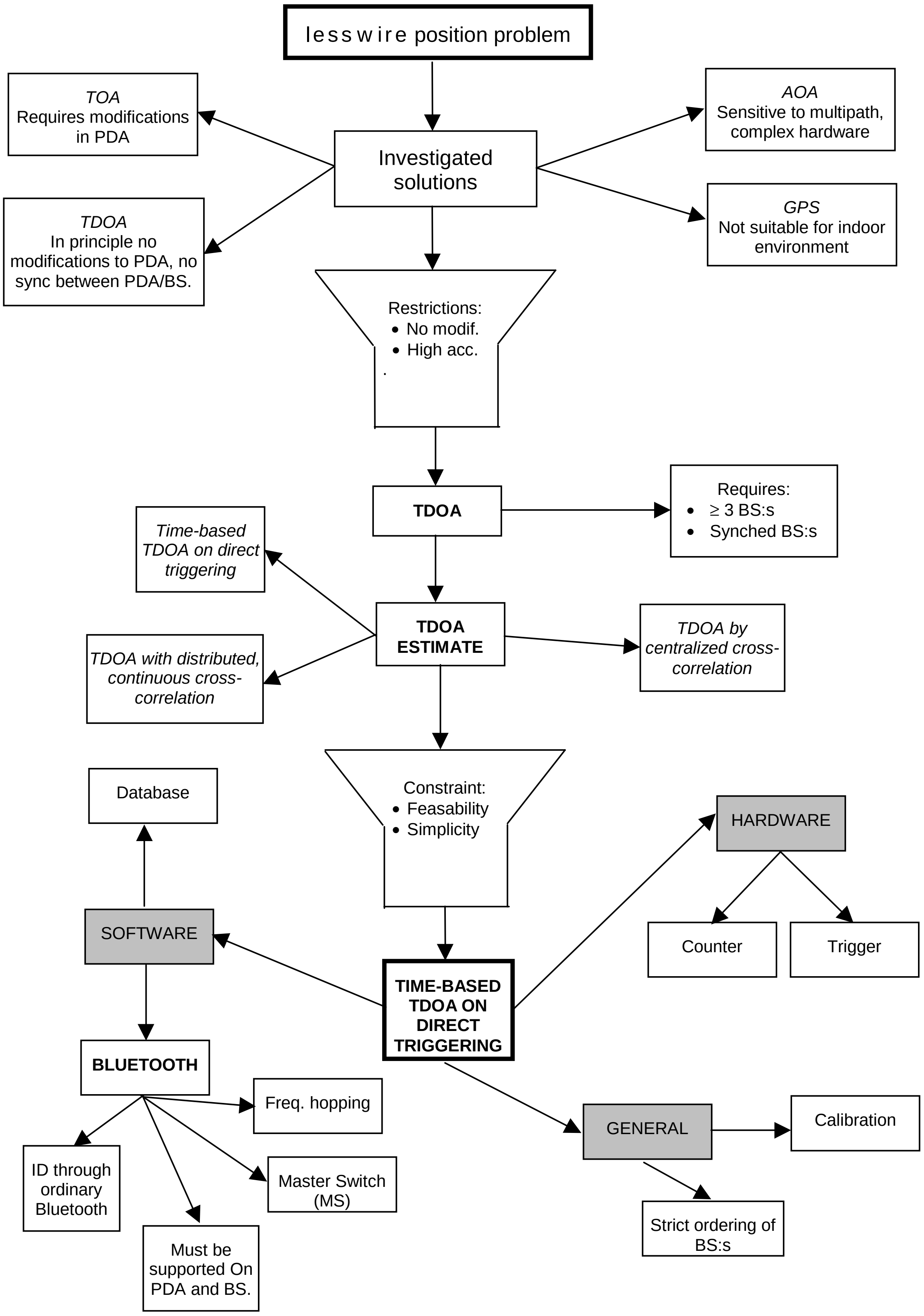

lesswire position problem
TOA
Requires modifications in PDA
TDOA
In principle no modifications to PDA, no sync between PDA/BS.
Investigated solutions
AOA
Sensitive to multipath, complex hardware
GPS
Not suitable for indoor environment
Restrictions:
• No modif.
• High acc.
TDOA
Requires:
• ≥ 3 BS:s
• Synched BS:s
Time-based TDOA on direct triggering
TDOA with distributed, continuous cross-correlation
TDOA ESTIMATE
TDOA by centralized cross-correlation
Constraint:
• Feasability
• Simplicity
Database
SOFTWARE
HARDWARE
Counter
Trigger
TIME-BASED TDOA ON DIRECT TRIGGERING
BLUETOOTH
Freq. hopping
ID through ordinary Bluetooth
Master Switch (MS)
Must be supported On PDA and BS.
GENERAL
Calibration
Strict ordering of BS:s

Figure 26: Decision tree for the proposed system design

5.2. General considerations

5.2.1. Base station setup

In principle, the base station setup could be performed in two ways: either we let the customers "randomly" (i.e. as they like) place the base stations. This turns out to be impossible for various practical reasons: first of all, we need to now the *exact* position of the base stations, which will involve high-precision measurements. Second: to guarantee position location capabilities, every spot needs to be covered by at least three base stations. Third: they must not be moved after the positioning. Hence, to be able to guarantee a position, the base stations must be arranged by the provider (i.e. by lesswire or under supervision of lesswire). By doing so, we know the position of each and we can insure that they will not be moved (for example assuming a typical fixed mounting in the ceiling).

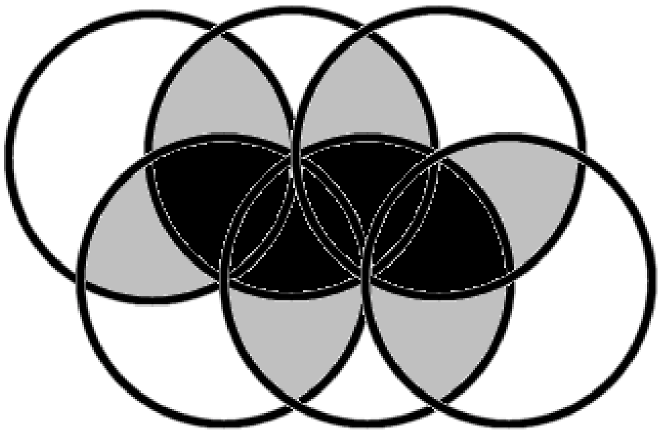


Figure 27: Optimal placement of base stations

Figure 27 shows an optimal placement of the base stations. The white sections are covered by one base station, the gray ones by two and finally; the black sections are covered by three stations, which is the minimum amount needed for the 2D TDOA. It is also required that the base stations are mounted in an absolute flat manner for the 2D-case.

Every base station contributes (for full coverage) to 1/6 of the entire, circular coverage area. That is, if circle perimeter is r, yet another base station will increase the completely (three base stations) covered area by $\pi r^2/6$. In the Bluetooth case, that translates to $\pi 10^2/6 \approx 52$ m^2.

The reasoning here is based on a model, and the real-world coverage areas are most likely to have irregular shapes due to environmental impact from furniture, building structures, etc. Still, more base stations would be desirable to improve system performance, and one suggestion is to provide a minimum structure like the one described, and let the customers supplement it by randomly placing their "own" base stations as they whish.

The base stations use regular Ethernet (or whatever standard will be used) for intercommunication.

5.2.2. Calibration

Due to the fact that every component[18] possesses unique characteristics, the system will have to be calibrated not only at installation but probably also after some usage due to environmental changes. If calibration can not compensate for the position errors that may occur, we are dealing with a disruptive environment and/or components with unacceptably large mean deviation.

[18] By "component" we refer to all hardware involved in the position location and synchronization.

5.3. Hardware

As discussed in section 3.3.1.1, the hardware needed (except for synchronization) is a high-performing counter and a trigger. The counter could be implemented simply as sequence of flip-flops (page 27) and must be driven by a synchronized (to the common time base) GHz-pulse generator.

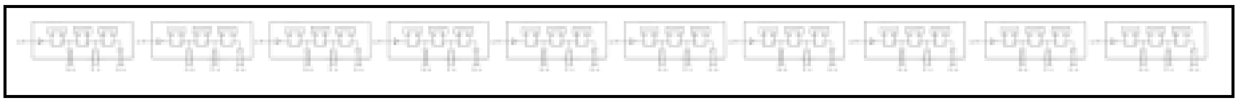


Figure 28: A cascade of flip-flops

We preferably do not want to stop the counter, but rather read-off the number and store it for processing. By doing this, we are able to process several subsequent requests and we do not have to reset the counters as we would have to after stopping them. To support this method, we need to supply a register capable of performing a snapshot on the counter status.

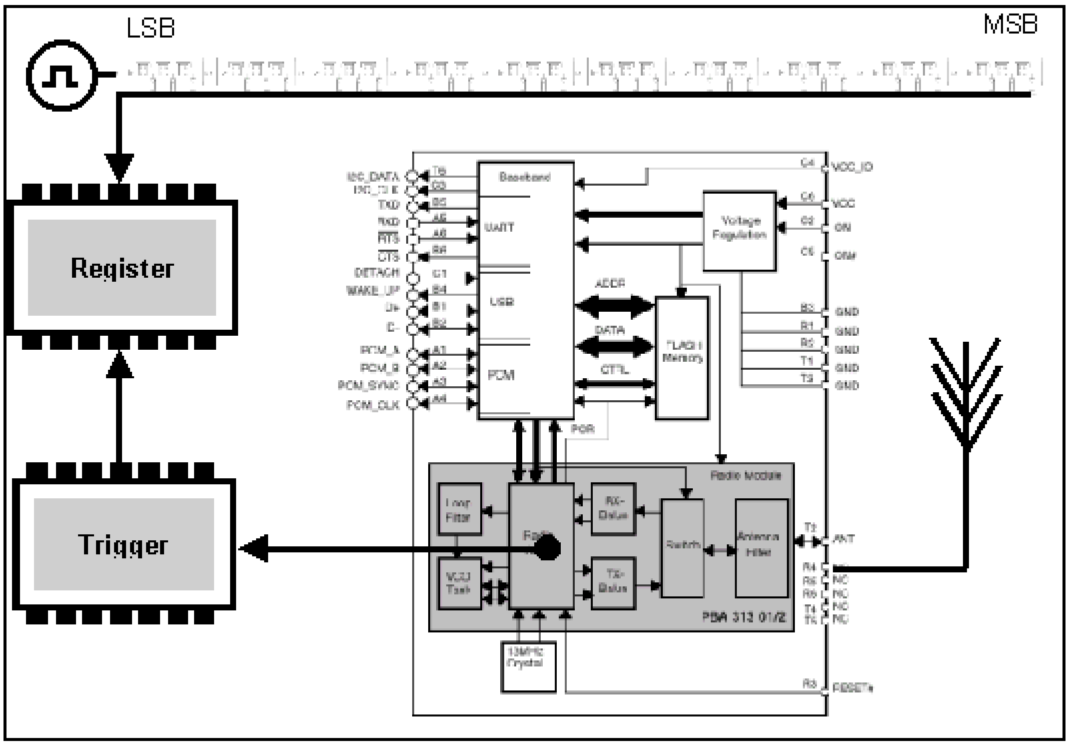


Figure 29: Hardware system design

The topmost part (from left to right) shows the pulse generator driving the counter with the least significant bit closest to the generator. The large, integrated bottom-right block shows an example of a fully integrated Bluetooth chip[19] together with antenna output.

Additional hardware is most likely to be required, but the key components behind this idea are shown in Figure 29. A matching mechanism, that keeps track of the local time and matches the time (found in the register) with the address of the sending PDA in the ordinary Bluetooth bit-stream at the base station side, will be needed.

[19] See http://www.ericsson.se/microe/bluetooth.html for a complete description of the Ericsson Bluetooth Module.

5.4. Software and work flow – putting it all together

After the previous discussions on general considerations, hardware requirements, the algebraic treatment where Chan's method was chosen, we can list the desired prerequisites.

5.4.1. Infrastructure prerequisites

- A complete, synchronized infrastructure consisting of Bluetooth based base stations.
- The base stations must be able to communicate to the central server in which position calculation is performed.
- Every spot (for 2D) on which position calculation is desired must be covered by three base stations according to the pattern described on page 61.
- Every base station must be equipped with additional hardware that implements the proposed scheme (page 63).
- Software wise (at application level), the base stations must be able to respond to a position request and force the base station to become a slave, participating in the piconet created by the PDA.

5.4.2. PDA prerequisites

- Bluetooth compatible PDAs are assumed.
- A software plug-in is needed that, upon a position request from the user, forces the PDA to become a master and broadcast to find as many slaves (i.e. base stations) as possible. It is important to make sure these slaves are base stations and not other PDAs. On a programmatic level, we will be able to select which slaves we want to be members of the piconet. In other words: "throw out" the ones which are not base stations.

5.4.3. Work flow

In this section, we will describe a typical scenario where a user wants to know his/her position, given the prerequisites in the previous section. This could of course be caused by the user requesting a service that in turn requires the position.

1. The master PDA initiates a position request.
2. The software plug-in forces the master PDA to become a master (if that is not the case) by means of the earlier described Slave-To-Master Switch (see [6], section 10.9.3).
3. According to [6], a channel definition (i.e. the creation of a Bluetooth channel) is performed by either paging or inquiring and the hopping sequence is derived from the BD_ADDR. This is very crucial to our solution: through this procedure, we can record the address of the requesting PDA. The PDA strives to create as big a piconet as possible, but a time-out must set a time limit on the scanning. At least three base stations are needed for the positioning in the general case. As mentioned before, we must make sure that the participating slaves are base stations and not other PDAs. This can be achieved at a programmatic level, where we simply lock out the unwanted PDAs from the piconet. The base stations must be able to identify themselves as base stations.
4. When at least three base stations are members of the newly created piconet, a zero-addressed broadcast message is being sent to the base stations.
5. The base stations read off the respective counters, make the time/address matching and send the pair to the main server.
6. On the server side, data from the base stations are collected and the position is calculated. *If* a position has been calculated before, a comparison is made (according to the criteria discussed on page 36) and if the new position "makes sense" (considering the criteria), we return the new position to the requesting PDA. Otherwise, we return an error message.

After 6 above, a new position request may be made by the same PDA or by another one.

A complex system like this possesses statistical behavior and there are several sources of errors. Therefore, we have intentionally left detailed data such as time-out values, triggering thresholds, etc. These must be empirically determined depending on the environment. Some environments may be highly disruptive and several requests must be performed to get an accurate position. This, in turn, will have an impact on time-out values, the number of participating base stations, etc. In some environments, the objects may be fast moving (hospital beds on wheels for instance), which requires another selection of parameters.

6. Discussion and future work

A work like this opens up many fields to be further investigated, and it is impossible to cover them all in detail within 100 pages. Considering that, this paper tends to be somewhat superficial, but considering the key goal: to answer the question stated by lesswire: *Can we do this at all, considering our constraints, and how could it be done?*, our hope is that these questions have been answered. The answer is: yes, it can be done, providing the appropriate hardware. To answer the question *How?*, we have given a system proposal that hopefully will serve as a guide for further work and/or implementation. It is our belief, that no purely theoretical paper can suggest a foolproof system that will work as predicted. No doubt, one should possess a great deal of humbleness towards the complexity of the problem and realize that practical field trials must be performed, hand in hand with the theoretical discussion. This research was purely theoretical due to the limited time, and it does not claim to be foolproof.

To summarize what can be done:

1. Method for calculating the position. Interesting for instance, would be to study hybrid systems, using both TDOA and AOA technology. As the timing of the Bluetooth PDA (or whatever technology comes with the PDA) gets better, the other methods may well be suitable. One thing is for sure: we can never escape the fact that we are dealing with radio signals, which propagate close to the speed of light. That single fact will forever have an impact on our system design.
2. Solving the equations. No matter which method you use for calculating the position, you will most likely end up with complex mathematics, due to the geometrical nature of the problem. There has been notable effort in this area, and a variety of algorithms has been proposed. Efficiency and accuracy are two key parameters for selecting the most appropriate algorithm.

3. Hardware / software. How to actually implement the solution in hardware / software. This is where the financial aspects come in. A trade-off must be made between accuracy and hardware / software. These two areas are both subject to rapid changes in technology and performance – and also price. What is not economically feasible today may well be feasible tomorrow.

We would like to encourage anyone to continue the work that has been initiated in this paper. The lesswire problem is interesting and challenging primarily because of its hard constraints and indoor application. Thousands and thousands of indoor square-meters can not provide positioning, and hence, they are potential sites for lesswire. Considering this, the market potential is huge.

7. Conclusions

In this thesis, we have concluded that, given the prerequisites:

1. High accuracy (67%) and high position resolution (+/- 2 m).
2. Affordability (as low complexity and cost as possible)
3. No modifications to the PDA hardware.
4. Bluetooth (not actually a constraint, but rather a goal)

…we can implement such a system by using:

- Normal Bluetooth environment (as defined in [6]).
- A TDOA-approach with at least three participating base stations.
- A *Time-based TDOA on direct triggering* scheme. Other techniques may well be used, but this one benefits from Bluetooth and requires a minimum of extra hardware.
- Chan’s method for solving the hyperbolic equations.

We believe the biggest challenge is actually to find a proper synchronization down, below 8 ns, preferably less. As it seems right now, the system is *not* economically feasible due to the hard requirements on the hardware and the amount of base stations needed to cover all spots where position location is desired.

References

[1] Ruth, Robert, DARPA Information Technology Office, http://www.darpa.mil/ito/research/glomo/index.html.

[2] Ioannidis, J., Duchamp, D., Maguire, G., *IP-based protocols for mobile internetworking*, Proceedings of SIGCOMM '91, ACM, September 1991.

[3] Kaplan, Elliott D. *Understanding GPS – principles and applications.*

[4] Beadle, H.W.P., Maguire Jr., G.Q., Smith, M.T., *Smart Badge: It beeps, It flashes, It knows when you are hot and sweaty* , October 1997.

[5] Harter, A., Hopper, A.., *A Distributed Location System for the Active Office*, AT&T Laboratories Cambridge, 24a Trumpington Street, Cambridge CB2 1QA, England (ftp://ftp.uk.research.att.com/pub/docs/att/tr.94.1.pdf).

[6] Bluetooth Special Interest Group: Nord, L., Haartsen, J., Melin, T., et. al., *Specification of the Bluetooth System v1.0 Specification Volume 1 Core A*, http://www.bluetooth.com, July 26th 1999 .

[7] Verniero, P., Todd Whitman, C. et el., *Report on the New Jersey Wireless Enhances 9-1-1 System Trial January 22 to April 30, 1997 – The First 100 Days*, State of New Jersey – Division of State Police, June 16 1997.

[8] Motyka, R., Calabrese, L., Anderson, S., *900 MHz TruePosition*, http://attila.stevens-tech.edu/~rmotyka/senior/index.html

[9] Trueposition Inc., *Time difference of arrival technology for locating narrowband cellular signals*, http://www.trueposition.com/tdoa.htm, December 17th 1999.
Artech House Publishers (ISBN 0-89006-793-7), 1996.

[10] Gabber, E., Wool, A., *On Location-Restricted Services*, IEEE Network, November/December 1999.

[11] Young, Ray, *Technical And Standards Division – Wireless Positioning Techniques and Services*, National Communications Volume 5/No. 2, September 1998.

[12] Stilp, Louis A., *Examining the Coming Revolution in Location Services*, Trueposition Inc., http://www.trueposition.com/forum.htm, January 2000.

[13] Swedberg, Göran, *Ericsson's mobile location solution*, Ericsson Review No. 4, http://www.ericsson.se/review/pdf/19990406.pdf, 1999.

[14] Werb, Jay, Lanzl, Colin, *Designing a positioning system for finding things and people indoors*, IEEE Spectrum: The Practical Engineer, September 1998.

[15] Mills, D.L., *Unix kernel modifications for precision time synchronization*, Electrical Engineering Department Report 94-10-1, University of Delaware, October 1994, 24 pp. Abstract: PostScript | PDF, Body: PostScript | PDF Major revision and update of: Network Working Group Report RFC-1589, University of Delaware, March 1994. 31 pp. ASCII

[16] Mills, D.L., Sep-01-1985, *Network Time Protocol (NTP).* (Format: TXT=30723 bytes) (Obsoleted by RFC1059, RFC1119, RFC1305)

[17] Mills., D.L. *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI.* October 1996. (Format: TXT=48620 bytes) (Obsoletes RFC1769)

[18] Mills, D. L. University of Delaware, e-mail, 22 March 2000

[19] Mills, D. L. *Generic Nanosecond Kernel Timekeeping Support*, http://www.eecis.udel.edu/~mills/nanokernel/index.htm.

[20] Eidson et al., *Method for recognizing events and synchronizing clocks (US Patent no. 5,566,180)*, Hewlett-Packard Company, Oct. 15, 1996

[21] Kopetz, H., *Loosely coupled distributed computer system with node synchronization for precision in real time applications*, US Patent no. 4,866,606, Austria Mikrosystem International GmbH, Sept. 12, 1989

[22] Hosgood, UK Patent no. 2,254,455A.

[23] Pierschel, M., Institute for High Performance microelectronics (IHP), personal communication and E-mail, 30 March 2000

[24] Fischer, G., Institute for High Performance microelectronics (IHP), personal communication, 23 March 2000

[25] Heyde, G., BFI Optilas GMBH, personal communication, 3 April 2000

[26] Sklar, Bernard, *Digital Communications – Fundamentals and Applications*, P T R Prentice Hall (ISBN 0-13-211-939-0), 1988

[27] Rappaport, T. S., Reed, J. H., Woerner, B.D., *Position Location Using Wireless Communications on Highways of the Future,* IEEE Communications Magazine, October 1996

[28] Gardner, W.A.; Spooner, C.M., *Detection and source location of weak cyclostationary signals: simplifications of the maximum-likelihood receiver,* Communications, IEEE Transactions on , Volume: 41 Issue: 6 , June 1993, Page(s): 905 -916

[29] Gardner, W.A.; Spooner, C.M., *Comparison of autocorrelation and cross-correlation methods for signal-selective TDOA estimation***,** *Signal Processing, IEEE Transactions on ,* Volume: 40 Issue: 10 , Oct. 1992, Page(s): 2606 -2608

[30] Chen, C.-K.; Gardner, W.A.**,** *Signal-selective time-difference of arrival estimation for passive location of man-made signal sources in highly corruptive environments. II. Algorithms and performance,* Signal Processing, IEEE Transactions on , Volume: 40 Issue: 5 , May 1992, Page(s): 1185 -1197

[31] Gardner, W.A.; Chen, C.-K., *Signal-selective time-difference-of-arrival estimation for passive location of man-made signal sources in highly corruptive environments. I. Theory and method,* Signal Processing, IEEE Transactions on , Volume: 40 Issue: 5 , May 1992, Page(s): 1168 -1184

[32] Xiang Yuan, Y.; Carter, G.C.; Salt, J.E., *Correlation among time difference of arrival estimators and its effect on localization in a multipath environment,* Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on , Volume: 5 , 1995, Page(s): 3163 -3166 vol.5

[33] Fong, G, Gardner, W. A., Schell, S. V., *An algorithm for Improved Signal-Selective Time-Difference Estimation for Cyclostationary Signals*, IEEE Signal Processing Letters Vol. 1, NO. 2, February 1994.

[34] Kayata Wesel, Ellen, *Wireless Multimedia Communications – Networking Video, Voice and Data*, Addison-Wesley Communication Series (ISBN 0-201-63394-9), 1998

[35] Knapp, C. H., Carter, G.C., *The Generalized Correlation Method for Estimation of Time Delay*, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-24, no. 4, 1976

[36] Press, William H., Teukolsky, Saul A., Vetterling, William T., Flannery, Brian P. *Numerical Recepies in C – The Art of Scientific Computing.* Cambridge University Press (ISBN 0-521-43108-5), 1995.

[37] Foy, W.H., *Position-Location Solutions by Taylor-Series Estimation*, IEEE Transactions on Aerospace and Electronic Systems, vol. AES-12, pp. 187-194, March 1976

[38] Y.T. Chan and K.C. Ho, *A simple and efficient Estimator for Hyperbolic Location*, IEEE Transactions on Signal processing, vol. 42, no. 8, pp. 1905-1915, August 1994.

Appendix A

The Levenberg-Marquardt simulation was run on a Sun Sparc Ultra 1 with Matlab version 5.3.0.10183.

Simulation 1: Finding the 8 ns

Chan code

```
function A=runchansim(maxiter)
% A simulation package for Chan's method as part of thesis work "A
distributed,
% mobile positioning system for wireless, handheld devices".
% Final simulation run 2000-06-07 19:40.


x=[];
y=[];
z=[];

err=[];
frequencyclasses=[0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0];
accum=zeros(1,11);
counterrors=0;
total=0;
error1=0;
error2=0;
error3=0;
errormin=0;
errormax=0;
maxerr=0;


BS1X=0;
BS1Y=0;
BS2X=5;
BS2Y=8.7;
BS3X=10;
BS3Y=0;
r=1;

t=cputime;
flops(0);

for rc=-8:8
           for sc=-8:8
                                 for tc=-8:8
                      PDAX=5.0;
                      PDAY=4.0;
```

```
          DistXactBS1PDA=sqrt((PDAX-BS1X)^2+(PDAY-BS1Y)^2);
          DistXactBS2PDA=sqrt((PDAX-BS2X)^2+(PDAY-BS2Y)^2);
          DistXactBS3PDA=sqrt((PDAX-BS3X)^2+(PDAY-BS3Y)^2);

          if DistXactBS1PDA<10.1 & DistXactBS2PDA<10.1 &
DistXactBS3PDA<10.1

                    error1=rc*1e-9;
                    error2=sc*1e-9;
                    error3=tc*1e-9;

          if errormin>min([error1,error2,error3])
                    errormin=min([error1,error2,error3]);
          end
          if errormax<max([error1,error2,error3])
                    errormax=max([error1,error2,error3]);
          end

          DistXactBS1PDA=DistXactBS1PDA+error1*300e6;
          DistXactBS2PDA=DistXactBS2PDA+error2*300e6;
          DistXactBS3PDA=DistXactBS3PDA+error3*300e6;
          R21=DistXactBS2PDA-DistXactBS1PDA;
          R31=DistXactBS3PDA-DistXactBS1PDA;

%***************CHAN START


X21=BS2X-BS1X;
X31=BS3X-BS1X;
Y21=BS2Y-BS1Y;
Y31=BS3Y-BS1Y;

K1=BS1X^2+BS1Y^2;
K2=BS2X^2+BS2Y^2;
K3=BS3X^2+BS3Y^2;

A = -[X21 Y21;X31 Y31]^(-1);
R = [R21;R31];
B = 0.5*[R21^2-K2+K1;R31^2-K3+K1];

RN=A*R;
BN=A*B;

BIGMAT=[1 (-2*(RN(1)*BN(1)+RN(2)*BN(2))/(1-RN(1)^2-RN(2)^2)) -
(BN(1)^2+BN(2)^2)/(1-RN(1)^2-RN(2)^2)];

R1=(-BIGMAT(2)+sqrt(BIGMAT(2)^2-
4*BIGMAT(1)*BIGMAT(3)))/(2*BIGMAT(1));

X=A*(R*R1+B);

%***************CHAN END
```

```
                      total=total+1;

                      x(r)=r;
                      y(r)=sqrt((X(1)-PDAX)^2+(X(2)-PDAY)^2);


           if y(r)>=frequencyclasses(1) & y(r)<frequencyclasses(2)
                      accum(1)=accum(1)+1;
           end
           if y(r)>=frequencyclasses(2) & y(r)<frequencyclasses(3)
                      accum(2)=accum(2)+1;
           end
           if y(r)>=frequencyclasses(3) & y(r)<frequencyclasses(4)
                      accum(3)=accum(3)+1;
           end
           if y(r)>=frequencyclasses(4) & y(r)<frequencyclasses(5)
                      accum(4)=accum(4)+1;
           end
           if y(r)>=frequencyclasses(5) & y(r)<frequencyclasses(6)
                      accum(5)=accum(5)+1;
           end
           if y(r)>=frequencyclasses(6) & y(r)<frequencyclasses(7)
                      accum(6)=accum(6)+1;
           end
           if y(r)>=frequencyclasses(7) & y(r)<frequencyclasses(8)
                      accum(7)=accum(7)+1;
           end
           if y(r)>=frequencyclasses(8) & y(r)<frequencyclasses(9)
                      accum(8)=accum(8)+1;
           end
           if y(r)>=frequencyclasses(9) & y(r)<frequencyclasses(10)
                      accum(9)=accum(9)+1;
           end
           if y(r)>=frequencyclasses(10) & y(r)<frequencyclasses(11)
                      accum(10)=accum(10)+1;
           end
           if y(r)>=frequencyclasses(11)
                      accum(11)=accum(11)+1;
           end
   end
 r=r+1;
end
end
end
```

```
display('************************************************')
display('MIN and MAX calculated error in meters:')
[min(y) max(y)]
display('Number of evaluated spots:')
total
display('CPU-time/calculation:')
(cputime-t)/sum(accum)
display('MIN and MAX introduced errors in nanoseconds (per base
station):')
[errormin errormax]
display('Fraction of measurements with an error less than 2
meters:')
(accum(1)+accum(2)+accum(3)+accum(4))/sum(accum)
display('Number of flops:')
flops
display('************************************************')
scatter (frequencyclasses,accum);
```

Chan - Output

```
>> runchanworst

ans =

*************************************************


ans =

MIN and MAX calculated error in meters:


ans =

         0    4.9534


ans =

Number of evaluated spots:


total =

        4913


ans =

CPU-time/calculation:


ans =

    0.0025


ans =

MIN and MAX introduced errors in nanoseconds (per base station):


ans =

   1.0e-08 *

   -0.8000    0.8000


ans =
Fraction of measurements with an error less than 2 meters:

ans =

    0.6743

ans =

Number of flops:

ans =

      962976

ans =

************************************************
```

Levenberg-Marquardt - Code

```
function A=runfsolve(method)
global x
global y
global accum
%Runs the fsolve algorithm with algorithm given in method

options=[];
options(18)=0;
options(10)=0;
options(1)=0;

x=[];
y=[];
z=[];

err=[];
frequencyclasses=[0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0];
accum=zeros(1,11);
counterrors=0;
total=0;
error1=0;
error2=0;
error3=0;
errormin=0;
errormax=0;


BS1X=0;
BS1Y=0;
BS2X=5;
BS2Y=8.7;
BS3X=10;
BS3Y=0;
INITIALGUESS=[5 4.3];
MAXITER=1500;
r=1;
rc=1;
sc=1;
tc=1;

t=cputime;
flops(0);


for rc=-8:8
           for sc=-8:8
                      for tc=-8:8

                      PDAX=5;
                      PDAY=4;
          DistXactBS1PDA=sqrt((PDAX-BS1X)^2+(PDAY-BS1Y)^2);
          DistXactBS2PDA=sqrt((PDAX-BS2X)^2+(PDAY-BS2Y)^2);
          DistXactBS3PDA=sqrt((PDAX-BS3X)^2+(PDAY-BS3Y)^2);
```

```
		if DistXactBS1PDA<10.1 & DistXactBS2PDA<10.1 &
DistXactBS3PDA<10.1
			TimeXactBS1PDA=DistXactBS1PDA/300e6;
			TimeXactBS2PDA=DistXactBS2PDA/300e6;
			TimeXactBS3PDA=DistXactBS3PDA/300e6;

			error1=rc*1e-9;
			error2=sc*1e-9;
			error3=tc*1e-9;

		if errormin>min([error1,error2,error3])
			errormin=min([error1,error2,error3]);
		end
		if errormax<max([error1,error2,error3])
			errormax=max([error1,error2,error3]);
		end

			TimeXactBS1PDA=TimeXactBS1PDA+error1;
			TimeXactBS2PDA=TimeXactBS2PDA+error2;
			TimeXactBS3PDA=TimeXactBS3PDA+error3;



			d21=TimeXactBS2PDA-TimeXactBS1PDA;
			d13=TimeXactBS1PDA-TimeXactBS3PDA;
			d23=TimeXactBS2PDA-TimeXactBS3PDA;
			i=[BS1X BS1Y BS2X BS2Y BS3X BS3Y d21 d13
d23];



		A=fsolve('tdoacalc',INITIALGUESS,options,[],i);

			total=total+1;

			x(r)=r;
			y(r)=sqrt((A(1)-PDAX)^2+(A(2)-PDAY)^2);

		if y(r)>=frequencyclasses(1) & y(r)<frequencyclasses(2)
			accum(1)=accum(1)+1;
		end
		if y(r)>=frequencyclasses(2) & y(r)<frequencyclasses(3)
			accum(2)=accum(2)+1;
		end
		if y(r)>=frequencyclasses(3) & y(r)<frequencyclasses(4)
			accum(3)=accum(3)+1;
		end
		if y(r)>=frequencyclasses(4) & y(r)<frequencyclasses(5)
			accum(4)=accum(4)+1;
		end
		if y(r)>=frequencyclasses(5) & y(r)<frequencyclasses(6)
			accum(5)=accum(5)+1;
		end
		if y(r)>=frequencyclasses(6) & y(r)<frequencyclasses(7)
			accum(6)=accum(6)+1;
		end
		if y(r)>=frequencyclasses(7) & y(r)<frequencyclasses(8)
```

```
                    accum(7)=accum(7)+1;
          end
          if y(r)>=frequencyclasses(8) & y(r)<frequencyclasses(9)
                    accum(8)=accum(8)+1;
          end
          if y(r)>=frequencyclasses(9) & y(r)<frequencyclasses(10)
                    accum(9)=accum(9)+1;
          end
          if y(r)>=frequencyclasses(10) & y(r)<frequencyclasses(11)
                    accum(10)=accum(10)+1;
          end
          if y(r)>=frequencyclasses(11)
                    accum(11)=accum(11)+1;
          end
     r=r+1;
    end
    end
end
end

display('************************************************')
display('MIN and MAX calculated errors in meters:')
[min(y) max(y)]
display('Number of evaluated spots:')
total
display('CPU-time/calculation:')
(cputime-t)/sum(accum)
display('MIN and MAX introduced errors in seconds (per base
station):')
[errormin errormax]
display('Fraction of measurements with an error less than 2
meters:')
(accum(1)+accum(2)+accum(3)+accum(4))/sum(accum)
display('Number of flops:')
flops
display('************************************************')
scatter (frequencyclasses,accum);
```

Levenberg-Marquardt – Output

```
>> runfworst

ans =

************************************************


ans =

MIN and MAX calculated errors in meters:


ans =

    0.0000    4.9534


ans =

Number of evaluated spots:


total =

        4913


ans =

CPU-time/calculation:


ans =

    0.0472


ans =

MIN and MAX introduced errors in seconds (per base station):


ans =

   1.0e-08 *

   -0.8000    0.8000


ans =
Fraction of measurements with an error less than 2 meters:

ans =

    0.6743

ans =

Number of flops:

ans =

    15946132

ans =

************************************************
```

Simulation 2: The system in practice

Chan code

```
function A=runchansim(maxiter)
% A simulation package for Chan's method as part of thesis work "A
distributed,
% mobile positioning system for wireless, handheld devices", by
Fredrik Christiansson.

x=[];
y=[];
z=[];

err=[];
frequencyclasses=[0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0];
accum=zeros(1,11);
counterrors=0;
total=0;
error1=zeros(1,maxiter);
error2=zeros(1,maxiter);
error3=zeros(1,maxiter);
errormin=0;
errormax=0;


BS1X=0;
BS1Y=0;
BS2X=5;
BS2Y=8.7;
BS3X=10;
BS3Y=0;


t=cputime;
flops(0);
```

```
for r=1:maxiter
                      randn('state',sum(100*clock));
                      PDAX=rand*BS3X;
                      PDAY=rand*BS2Y;

           DistXactBS1PDA=sqrt((PDAX-BS1X)^2+(PDAY-BS1Y)^2);
           DistXactBS2PDA=sqrt((PDAX-BS2X)^2+(PDAY-BS2Y)^2);
           DistXactBS3PDA=sqrt((PDAX-BS3X)^2+(PDAY-BS3Y)^2);

if DistXactBS1PDA<10.1 & DistXactBS2PDA<10.1 & DistXactBS3PDA<10.1
                      error1(r)=randn*5e-9;
                      error2(r)=randn*5e-9;
                      error3(r)=randn*5e-9;

           if errormin>min([error1(r),error2(r),error3(r)])
                      errormin=min([error1(r),error2(r),error3(r)]);
           end
           if errormax<max([error1(r),error2(r),error3(r)])
                      errormax=max([error1(r),error2(r),error3(r)]);
           end


           DistXactBS1PDA=DistXactBS1PDA+error1(r)*300e6;
           DistXactBS2PDA=DistXactBS2PDA+error2(r)*300e6;
           DistXactBS3PDA=DistXactBS3PDA+error3(r)*300e6;

           R21=DistXactBS2PDA-DistXactBS1PDA;
           R31=DistXactBS3PDA-DistXactBS1PDA;

%*****************CHAN START

X21=BS2X-BS1X;
X31=BS3X-BS1X;
Y21=BS2Y-BS1Y;
Y31=BS3Y-BS1Y;

K1=BS1X^2+BS1Y^2;
K2=BS2X^2+BS2Y^2;
K3=BS3X^2+BS3Y^2;

A = -[X21 Y21;X31 Y31]^(-1);
R = [R21;R31];
B = 0.5*[R21^2-K2+K1;R31^2-K3+K1];

RN=A*R;
BN=A*B;

BIGMAT=[1 (-2*(RN(1)*BN(1)+RN(2)*BN(2))/(1-RN(1)^2-RN(2)^2)) -
(BN(1)^2+BN(2)^2)/(1-RN(1)^2-RN(2)^2)];

R1=(-BIGMAT(2)+sqrt(BIGMAT(2)^2-
4*BIGMAT(1)*BIGMAT(3)))/(2*BIGMAT(1));

X=A*(R*R1+B);

%*****************CHAN END
```

```
                      total=total+1;

                      x(r)=r;
                      y(r)=sqrt((X(1)-PDAX)^2+(X(2)-PDAY)^2);

           if y(r)>=frequencyclasses(1) & y(r)<frequencyclasses(2)
                      accum(1)=accum(1)+1;
           end
           if y(r)>=frequencyclasses(2) & y(r)<frequencyclasses(3)
                      accum(2)=accum(2)+1;
           end
           if y(r)>=frequencyclasses(3) & y(r)<frequencyclasses(4)
                      accum(3)=accum(3)+1;
           end
           if y(r)>=frequencyclasses(4) & y(r)<frequencyclasses(5)
                      accum(4)=accum(4)+1;
           end
           if y(r)>=frequencyclasses(5) & y(r)<frequencyclasses(6)
                      accum(5)=accum(5)+1;
           end
           if y(r)>=frequencyclasses(6) & y(r)<frequencyclasses(7)
                      accum(6)=accum(6)+1;
           end
           if y(r)>=frequencyclasses(7) & y(r)<frequencyclasses(8)
                      accum(7)=accum(7)+1;
           end
           if y(r)>=frequencyclasses(8) & y(r)<frequencyclasses(9)
                      accum(8)=accum(8)+1;
           end
           if y(r)>=frequencyclasses(9) & y(r)<frequencyclasses(10)
                      accum(9)=accum(9)+1;
           end
           if y(r)>=frequencyclasses(10) & y(r)<frequencyclasses(11)
                      accum(10)=accum(10)+1;
           end
           if y(r)>=frequencyclasses(11)
                      accum(11)=accum(11)+1;
           end
           end
end
```

```
display('************************************************')
display('Number of valid positions:')
total
display('CPU-time/calculation:')
(cputime-t)/sum(accum)
display('MIN and MAX-errors in seconds:')
[errormin errormax]
display('Fraction of measurements with an error less than 2
meters:')
(accum(1)+accum(2)+accum(3)+accum(4))/sum(accum)
display('Number of flops:')
flops
display('Standard deviation on BS1:')
std(error1)
display('Standard deviation on BS2:')
std(error2)
display('Standard deviation on BS3:')
std(error3)
display('************************************************')
scatter (frequencyclasses,accum);
```

Chan output

```
>> runchansim(1500)

ans =

************************************************


ans =

Number of valid positions:


total =

        1094


ans =

CPU-time/calculation:


ans =

    0.0035


ans =

MIN and MAX-errors in seconds:


ans =

   1.0e-07 *

   -0.1700    0.1559


ans =

Fraction of measurements with an error less than 2 meters:


ans =

    0.6938


ans =

Number of flops:


ans =
      247047

ans =

Standard deviation on BS1:


ans =

   1.5829e-09


ans =

Standard deviation on BS2:


ans =

   4.1537e-09


ans =

Standard deviation on BS3:


ans =

   4.2729e-09


ans =

************************************************


ans =

         0   -0.1000
   -0.1149    0.0575

>>
```

Levenberg-Marquardt code

```
function A=runfsolve(maxiter)
global x
global y
global accum

options=[];
options(18)=0;
options(10)=0;
options(1)=0;

x=[];
y=[];
z=[];

err=[];
frequencyclasses=[0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0];
accum=zeros(1,11);
counterrors=0;
total=0;
error1=zeros(1,maxiter);
error2=zeros(1,maxiter);
error3=zeros(1,maxiter);
errormin=0;
errormax=0;

BS1X=0;
BS1Y=0;
BS2X=5;
BS2Y=8.7;
BS3X=10;
BS3Y=0;
INITIALGUESS=[5 4.3];

t=cputime;
flops(0);

for r=1:maxiter

                      randn('state', sum(100*clock));

                      PDAX=rand*BS3X;
                      PDAY=rand*BS2Y;

           DistXactBS1PDA=sqrt((PDAX-BS1X)^2+(PDAY-BS1Y)^2);
           DistXactBS2PDA=sqrt((PDAX-BS2X)^2+(PDAY-BS2Y)^2);
           DistXactBS3PDA=sqrt((PDAX-BS3X)^2+(PDAY-BS3Y)^2);

if DistXactBS1PDA<10.1 & DistXactBS2PDA<10.1 & DistXactBS3PDA<10.1
           TimeXactBS1PDA=DistXactBS1PDA/300e6;
           TimeXactBS2PDA=DistXactBS2PDA/300e6;
           TimeXactBS3PDA=DistXactBS3PDA/300e6;

                      error1(r)=randn*5e-9;
                      error2(r)=randn*5e-9;
                      error3(r)=randn*5e-9;
```

```
            if errormin>min([error1(r),error2(r),error3(r)])
                        errormin=min([error1(r),error2(r),error3(r)]);
            end
            if errormax<max([error1(r),error2(r),error3(r)])
                        errormax=max([error1(r),error2(r),error3(r)]);
            end

            TimeXactBS1PDA=TimeXactBS1PDA+error1(r);
            TimeXactBS2PDA=TimeXactBS2PDA+error2(r);
            TimeXactBS3PDA=TimeXactBS3PDA+error3(r);
            d21=TimeXactBS2PDA-TimeXactBS1PDA;
            d13=TimeXactBS1PDA-TimeXactBS3PDA;
            d23=TimeXactBS2PDA-TimeXactBS3PDA;

            i=[BS1X BS1Y BS2X BS2Y BS3X BS3Y d21 d13 d23];

            A=fsolve('tdoacalc',INITIALGUESS,options,[],i);

            total=total+1;
            x(r)=r;
            y(r)=sqrt((A(1)-PDAX)^2+(A(2)-PDAY)^2);

            if y(r)>=frequencyclasses(1) & y(r)<frequencyclasses(2)
                        accum(1)=accum(1)+1;
            end
            if y(r)>=frequencyclasses(2) & y(r)<frequencyclasses(3)
                        accum(2)=accum(2)+1;
            end
            if y(r)>=frequencyclasses(3) & y(r)<frequencyclasses(4)
                        accum(3)=accum(3)+1;
            end
            if y(r)>=frequencyclasses(4) & y(r)<frequencyclasses(5)
                        accum(4)=accum(4)+1;
            end
            if y(r)>=frequencyclasses(5) & y(r)<frequencyclasses(6)
                        accum(5)=accum(5)+1;
            end
            if y(r)>=frequencyclasses(6) & y(r)<frequencyclasses(7)
                        accum(6)=accum(6)+1;
            end
            if y(r)>=frequencyclasses(7) & y(r)<frequencyclasses(8)
                        accum(7)=accum(7)+1;
            end
            if y(r)>=frequencyclasses(8) & y(r)<frequencyclasses(9)
                        accum(8)=accum(8)+1;
            end
            if y(r)>=frequencyclasses(9) & y(r)<frequencyclasses(10)
                        accum(9)=accum(9)+1;
            end
            if y(r)>=frequencyclasses(10) & y(r)<frequencyclasses(11)
                        accum(10)=accum(10)+1;
            end
                        if y(r)>=frequencyclasses(11)
                                    accum(11)=accum(11)+1;
                        end

            end
end
```

```
display('************************************************')
display('Number of valid positions:')
total
display('CPU-time/calculation:')
(cputime-t)/sum(accum)
display('MIN and MAX-errors in seconds:')
[errormin errormax]
display('Fraction of measurements less than 2 meters:')
(accum(1)+accum(2)+accum(3)+accum(4))/sum(accum)
display('Number of flops:')
flops
display('Standard deviation on BS1:')
std(error1)
display('Standard deviation on BS2:')
std(error2)
display('Standard deviation on BS3:')
std(error3)
display('************************************************')
%scatter (frequencyclasses,accum);
```

Levenberg-Marquardt output

```
************************************************

ans =

Number of valid positions:

total =

        1108

ans =

CPU-time/calculation:

ans =

    0.0619

ans =

MIN and MAX-errors in seconds:

ans =

   1.0e-07 *

   -0.1697    0.1372

ans =

Fraction of measurements less than 2 meters:

ans =

    0.6823

ans =

Number of flops:

ans =

     4578287
```

```
ans =

Standard deviation on BS1:


ans =

   3.1258e-09


ans =

Standard deviation on BS2:


ans =

   3.4404e-09


ans =

Standard deviation on BS3:


ans =

   3.4567e-09


ans =

************************************************


ans =

    1.7467    1.0619

>>
```