Real-Time Object Tracking with Wireless Sensor Networks

Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Control Engineering

DIPLOMA THESIS

Luleå University of Technology

Department of Space Science

Kiruna Space Campus

by

Inderjit Singh

Prague, May 2007

Declaration

I ensure hereby that the thesis entitled “Real-Time Object Tracking with Wireless Sensor Networks” is independently written and used no different than the indicated aids under the guidance of Ing. Jiří Trdlička, Department of Control Engineering at Czech Technical University.

Inderjit Singh

Prague, May 2007

Abstract

The first attempt of the thesis work have been to implement a dynamic system with WSN that uses the Time Division Multiple Access (TDMA) with Extended Adaptive Slot Assignment Protocol (E-ASAP). It has been shown in this work that it is a very suitable protocol that can be used in a dynamic WSN. Although, the complete functionality of the system is not implemented due to the time restriction on the work, a continuous work have been suggested with the multi hop technique for passing data throughout the network. Only then the system is considered to be fully usable.

Is is also presented how localisation of a source, in this case light, can be traced by using the mentioned protocols above. The algorithm localising via averaging is chosen for this purpose and is proven to be well suited for this application. There was a resulted failure in accuracy while using this algorithm. This was showed to be mainly the lack of a proper calibration to the source that was overlooked during the test procedure. However, the recommendation of a proper calibration with source is described. Also an enhanced algorithm is suggested for improving the result of the source position.

Contents

Introduction

Distributed wireless sensor networks (WSNs) have been increasing in popularity for a wide range of applications. The main purpose of sensor networks is to monitor an area, including detecting, identifying, localising and tracking one or more objects of interest. As with any wireless communication in distributed systems, data collision is a major issue and need to be resolved when using WSNs. Another main issue is the energy consumption of each individual sensor. It is the scarcest resource that determines the lifetime of WSNs. WSNs are meant to be deployed in large numbers in various environments, including remote and hostile regions[10].

The aim of the thesis was to design and implement a wireless sensor network for object tracking under real-time constraints using time division multiple access (TDMA) with Extended Adaptive Slot Assignment Protocol (E-ASAP). The application is implemented in TinyOS operating system with nesC as the programming language using ZigBee technology for its low power consumption.

The document is structured in such that here in the introduction, the basic terminology is explained that is used throughout the whole document. Then in the second chapter, theoretical backgrounds are explained that are used for the realisation of the work. Further on, in chapter three the implementation of the complete work with in mind of the theoretical consideration, presented. And finally, in the fourth, the tests results are presented to show the performance of the system.

The remaining chapters concludes the work with the comments on future work for improving the system.

1.1 Wireless Sensor Networks (WSNs)

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices, also called as nodes, using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. The development of wireless sensor networks was originally motivated by military applications such as battlefield surveillance. However, they are now used in many civilian application areas, including environment and habitat monitoring, healthcare applications, home automation, and traffic control[10].

The applications for WSNs are many and varied. They are used in commercial and industrial applications to monitor data that would be difficult or expensive to monitor using wired sensors. They could be deployed in wilderness areas, where they would remain for many years (monitoring some environmental variables) without the need to recharge/ replace their power supplies. They could also form a perimeter about a property and monitor the progression of intruders (passing information from one node to the next). Etc. [10]

A typical wireless sensor contains, apart from its sensor(s), with a wireless communication device such as radio transceiver or similar, energy source (typically batteries) and a small micro-controller.

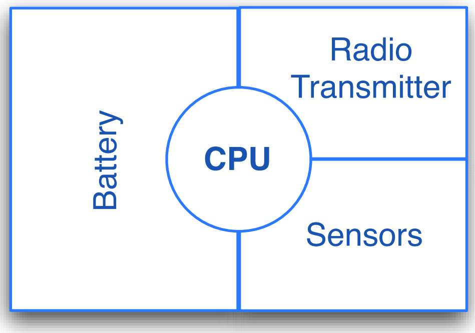

Figure 1.1.1 - A typical wireless sensor

The size of a single sensor node can vary from shoe-box size down to grain of dust. The cost of sensor nodes is similarly variable, ranging from hundreds of euros to a few cents, depending on the size of the sensor network and the complexity required of individual sensor node. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and bandwidth[10].

1.1.1 Zigbee

ZigBee is the name of a specification for a suite of high level communication protocols using small, low-power digital radios based on the IEEE 802.15.4 standard for wireless personal area networks (WPANs). ZigBee is targeted at RF applications that require a low data rate, long battery life, and secure networking. Its current focus is to define a general-purpose, inexpensive and self-organising mesh network[1] that can be used for various applications [11] as mentioned above.

1.1.2 TinyOS/nesC

Operating systems for wireless sensor network nodes typically are less complex than general-purpose operating systems both because of the special requirements of sensor network applications and because of the resource constraints in sensor network hardware platforms. Applications to these sensor networks are usually not interactive in the same way as applications for PCs. Because of this, the operating system does not need to include support for user interfaces. Furthermore, the resource constraints in terms of memory and memory mapping hardware support make mechanisms such as virtual memory either unneeded or impossible to implement[10].

TinyOS is an open source component-based operating system targeting WSNs and is developed by a consortium led by the University of California, Berkeley in cooperation with Intel Research[12].

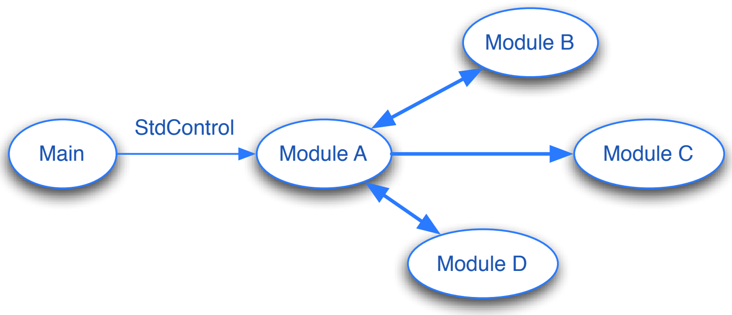

Figure 1.1.2.1 - Simple example of component-based architecture of TinyOS/nesC.

It is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes. It is designed to be able to incorporate rapid innovation as well as to operate within the severe memory constraints inherent in sensor networks. Eventually, it is intended to be incorporated into Smartdust, a hypothetical network of tiny

[1] *Mesh network* - A network whose nodes (devices) are connected to each other either directly or via nodes.

wireless microelectromechanical systems (MEMS) sensors, robots, or devices, installed with wireless communications, that can detect anything from light and temperature to vibrations [12].

nesC is a dialect of the C programming language, optimised for the memory limitations of sensor networks. The TinyOS code is statically linked with program code, and compiled into a small binary, using a custom GNU toolchain. It provides built-in interfaces, modules, and sensor-board specific configurations, which allows to build programs as a set of modules, which perform program-specific tasks. TinyOS modules themselves provide interface to the standard kinds of hardware inputs, outputs, and sensors[12].

1.2 Time Division Multiple Access (TDMA)

Time division multiple access (TDMA) is a channel access method for shared medium (usually radio) networks. It allows several users to share the same frequency channel by dividing the signal into different time slots. The devices transmit in rapid succession, one after the other, each using his own time slot, Figure 1.2.1. This allows multiple devices to share the same transmission medium (e.g. radio frequency channel)[1]. For maximum channel utilisation, slots are then combined into a TDMA period that are repeated continuously, Figure 1.2.2.

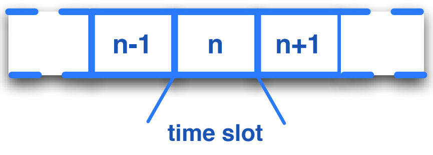

Figure 1.2.1 A slot within the TDMA

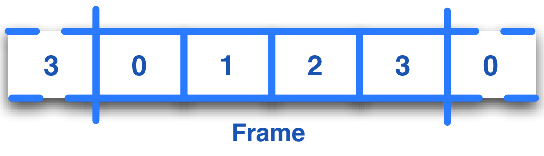

Figure 1.2.2 - Example of one TDMA period containing four slots.

In Ad hoc[2] sensor networks, each node continuously monitors the ambient environment and communicates with other nodes in its neighbouring nodes. This results into very heavy traffic load in the whole network. In order to handle such situation, TDMA is well suited for the solution due to its ability to provide collision free packet transmission regardless of the traffic load. Until now, many transmission scheduling protocols using TDMA applied to ad hoc networks have been proposed, however, most of them do not consider autonomous behaviour of mobile nodes where they cannot update slot assignment of each node enters or leaves the the network[1].

[2] *Ad hoc* - ("Spontaneous"). A network with temporary connections, in which the network devices are part of the network only for a duration of a communication session.

2

Theoretical considerations

To comply with the conditions required for realising the work, the theoretical considerations are divided in to three parts. Namely, a protocol suitable for ad-hoc network, a time synchronisation algorithm that can be easily implemented for that protocol, and finally, localisation of a source by using these two as the foundation.

For an ad-hoc network, Extended Adaptive Slot Assignment Protocol (E-ASAP) have been selected [1]. This protocol has the excellent capabilities for the maximum utilisation for an ad-hoc network and is well suited for this work.

Secondly, a time synchronisation algorithm must be present for making a dynamic network using the TDMA with E-ASAP protocol to work satisfyingly. It has been considered to choose a simple algorithm as possible that is both reliable and realisable. For this purpose, an simple averaging algorithm is used and described in chapter 2.2.

Finally, by using the these mentioned theories, for localising the source, localising via averaging [2].

2.1 Extended Adaptive Slot Assignment Protocol (E-ASAP)

In E-ASAP protocol, each node holds the information of its assigned slot and TDMA period as well as its neighbours' and hidden nodes'. A hidden node is defined by a node that is one hop away from the neighbour, Figure 2.1.1.

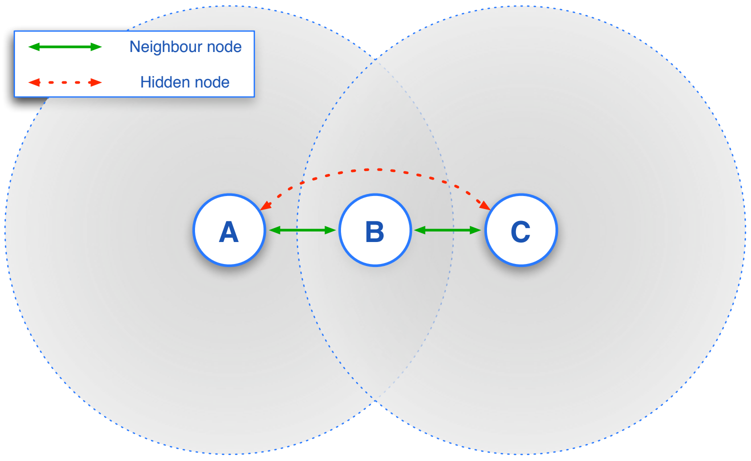

Figure 2.1.1 - Simple example of a neighbour and a hidden node.

Node A and B are neighbours to each other as B and C are. Due to the limitation of the transmission range of each node, the data that from node A to C needs to be passed through node B and thereby considered as hidden node relative to each other. Hence, one hop away from its neighbour.

The need for the information on hidden node is required for avoiding data collision. If for example node C joins the network after node A and B, it needs to know that node B has a neighbour A that sends its transmission on for example slot ***one***. For this reason, node C cannot assign slot ***one*** to itself for there will be a collision on node B at that slot.

2.1.1 TDMA period format in E-ASAP

Each node in the network has a TDMA period, or frame, defined where it presents how many slots there are within this period. In E-ASAP, this period is set as a power of two, although minimum size is fixed to four (slots). It can then be changed dynamically depending on the slot assignment in its contention area[3]. The first slot in the TDMA period, slot ***zero***, is reserved for all new nodes that enter the contention area.

[3] *Contention area* - All nodes that are in the vicinity within one hope away, i.e. neighbours and hidden nodes.

2.1.1.1 Simple TDMA period format example

Considering the following simple scenario shown in Figure 2.1.1, the information that is held by the node A will be then as following:

Own
Slot(s) / Frame
1 / 4

Neighbours	
Node	Slot(s) / Frame
B	2 / 4

Hidden nodes	
Node	Slot(s) / Frame
C	3 / 4

Figure 2.1.1.1.1 - Node A's information about contention area.

Or as seen with the TDMA representation:

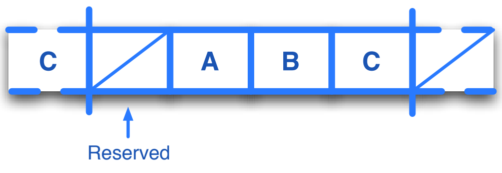

Figure 2.1.1.1.2 - E-ASAP presentation in TDMA.

Here, it is considered that nodes A, B and C have joined the network respectively in time.

2.1.1.2 A more complex WSN example

Lets consider an example of a more complex network with ten nodes.

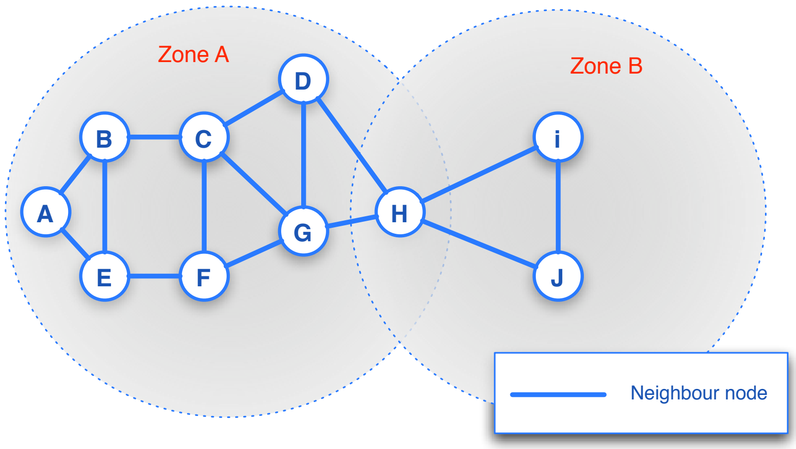

Figure 2.1.1.2.1 - Example of a more complex WSN

In Figure 2.1.1.2.1, node H sets its own TDMA period as eight slots and connects to nodes I and J whose TDMA period is four slots. Without any collision, it can transmit in slot two.

This is because the TDMA period of four slots can be easily translated to the period of eight slots by doubling the period. Following information is then held by node H:

Own
Slot(s) / Frame
2 / 8

Neighbours	
Node	Slot(s) / Frame
D	6 / 8
G	4 / 8
I	1 / 4
J	3 / 4

Hidden nodes	
Node	Slot(s) / Frame
C	1 / 8
F	5 / 8

Figure 2.1.1.2.2 - Information held by node H in E-ASAP

Or as seen with the TDMA representation:

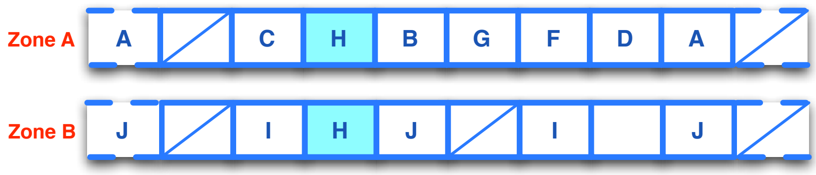

Figure 2.1.1.2.3 - TDMA representation of the network in Figure 1.3.1.2.1

This information, Figure 2.1.1.2.2, is called *Information packets*, INFs, and contains the assignment information about the sender and its neighbours that is held by the sender.

2.1.2 Slot assignment

To obtain the slot assignment information in the contention area, a new node first collects INFs transmitted by its neighbours. Then, the new node sets its TDMA period as four, which is the minimum allowed length in E-ASAP, and searches a slot which it assigns to itself via the following procedures.

1. *Getting an unassigned slot (GU)*

   If the new node finds the first empty slot in the current TDMA period, it assigns a slot to itself.

2. *Doubling the TDMA period (DF)*

   If no slot is available in the current TDMA period, the new node doubles the period and tries to assign a slot by GU. This procedure is repeated until the new node finds a slot. The TDMA period will then be a power of two but minimum four.

When the new node has joined the network, it will send its INF packet to the neighbourhood and all the nodes in the contention area will then adjust theirs accordingly.

Note that when there is a case where there are no slots available in the current TDMA period, the new node will double its period and will assign the first slot in the latter half to itself. That slot will be the slot zero for the neighbouring nodes and thereby will double their period as well. See the example in chapter 2.1.2.1 *Slot assignment example*.

2.1.2.1 Slot assignment example

Consider a network, Figure 2.1.2.1.1, where three nodes already are present in a network and where a new node, D, joins the network.

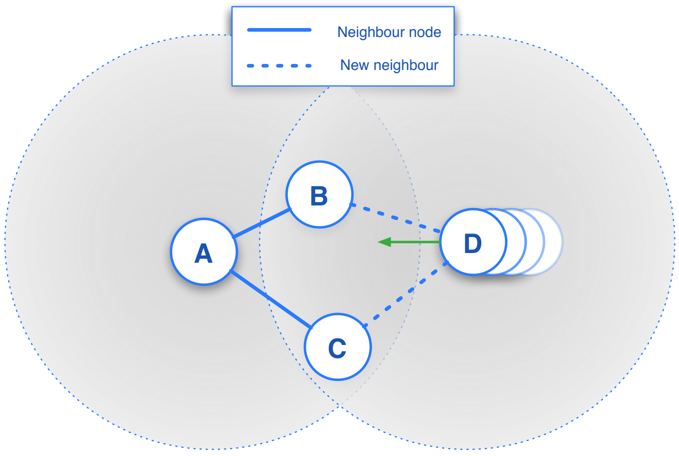

Figure 2.1.2.1.1 - Slot assignment of newly entered node D.

First, the node D collects the INFs sent be all nodes in the neighbourhood and will holds the following INF:

Own
Slot(s) / Frame
? / 4

Neighbours	
Node	Slot(s) / Frame
B	2 / 4
C	3 / 4

Hidden nodes	
Node	Slot(s) / Frame
A	1 / 4

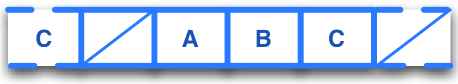

Figure 2.1.2.1.2 - INF gathered by the node D before joining the network.

Node D cannot find an empty slot in the current TDMA period and therefore doubles its period. Then it assigns the first slot in the latter half of the new period to itself and joins the network. The resulting INF on node D will be as following:

Own
Slot(s) / Frame
4 / 8

Neighbours	
Node	Slot(s) / Frame
B	2 / 8
C	3 / 8

Hidden nodes	
Node	Slot(s) / Frame
A	1 / 4
A	5 / 4

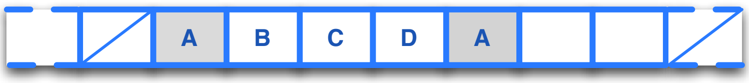

Figure 2.1.2.1.3 - INF held by node D after joining the network.

Note that, nodes B and C, but not A, have doubled their TDMA period to avoid collision. Node A does not need to double its period because there is no collision with network. Although, all the other nodes need to remember that node A will repeat its transmission at slot five in their period. From node A's point of view, it is always transmitting on slot one.

2.1.3 Releasing Slot assignment

When a node exits from the network, it releases its slot by stop transmitting INF packets. The neighbouring nodes can detect the exit when packets are no longer received from the exited node. After certain time, determined by their own TDMA period, the exited node will be assumed to have left the network. When detected the exit, the neighbouring nodes releases the slot from their slot assignment information. In this way, the operation of releasing slot assigned to the exited node is completed[1].

When an exit of a node have been detected and one of the following conditions is satisfied for each neighbour of the exited node, it halves its TDMA period and sends updated information to all nodes in its contention area[1].

1. The TDMA period of all neighbours are smaller than to itself.
2. The first slot in the latter half of the TDMA period is an unassigned slot, and one of the following conditions are satisfied. The first slot in the latter half of the period corresponds to slot zero that determines the doubling of the period.
   - The slot assignment information in the both halves of the period are identical.

- For all slots in the former half of the period and the corresponding slots in the latter half are either assigned to the same node or unassigned.

2.2 Time synchronisation

Synchronising the time between nodes is probably the most important characteristic for any distributed system, in particular WSNs using TDMA scheduling. To comply with easy algorithm and accurate time synchronisation between the nodes, an simple averaging algorithm is considered for this application. It is a simple operation of a node's local time over its neighbours. Each node computes the local average value directly when a INF packet is received from a neighbour containing the neighbour's current time value.

$$t_{local} = \frac{t_{remote} + t_{local}}{2} \tag{2.2.1}$$

2.2.1 Simulation on time synchronisation

Two simulations have been conducted using this method for two respectively six nodes as neighbours in a network.

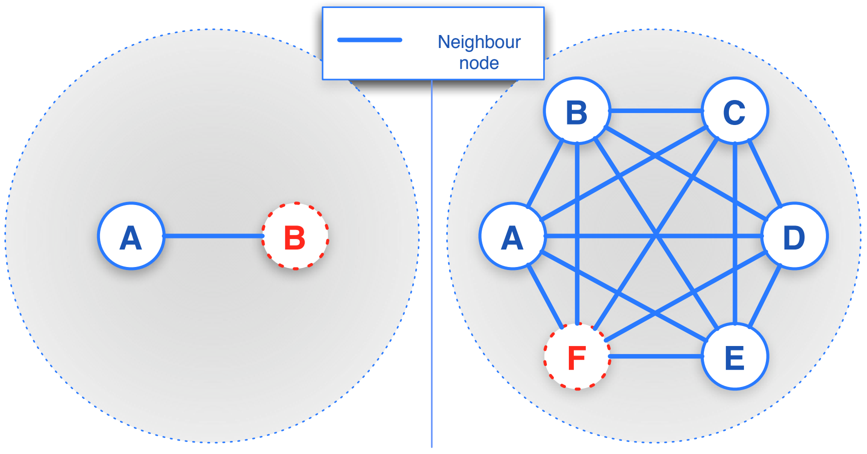

Figure 2.2.1.1 - Simulation with two respective six nodes as neighbours with injected error in node B respective F.

In this simulation, Figure 2.2.1.1, the time slot (***interval***) is set to 1000 ms seconds. The scenario of this simulation is that all the nodes have their slots designated accordingly by their letters, i.e A = 1, B = 2, C = 3 etc. Once all the nodes have joined the network, the node with the injected error of 500 ms, node B respective F, joins the network. By applying the averaging algorithm, following results were established:

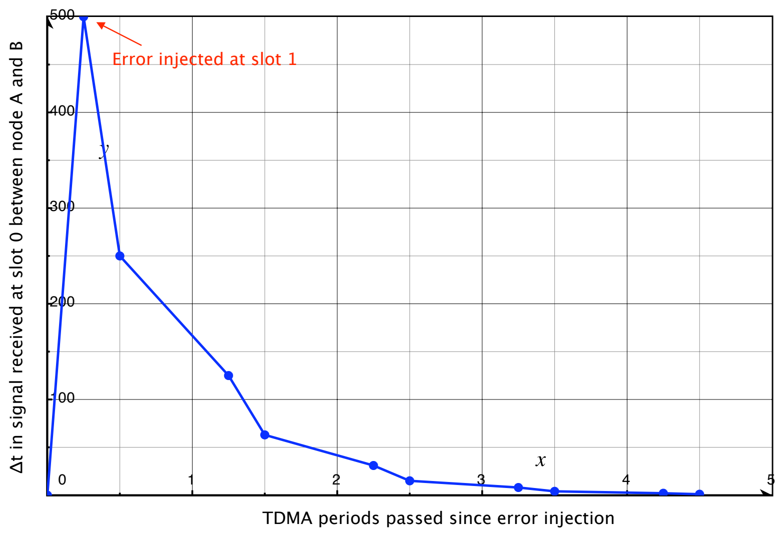

Figure 2.2.1.2 - Time difference between the nodes when injected error in node B at slot 1.

For two nodes in the network, it takes roughly five TDMA periods for the error to be corrected. This is because only two averaging calculations are made for each period, at slot one by the node A and slot two by the node B.

When conducted the same simulation for six nodes where the injected error is enforced on node F instead,

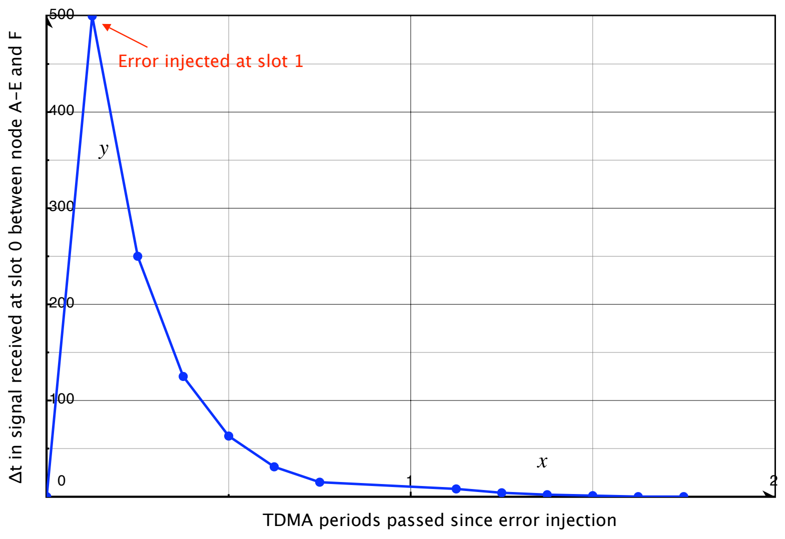

Figure 2.2.1.2 - Time difference between the nodes when injected error in node F at slot 1.

the error is corrected within two TDMA periods. Here, the time difference is averaged six times per period and therefore improves the result dramatically. Hence, the more nodes there are in the neighbourhood, the more quickly will the network adapt to errors.

2.3 Source localisation

To localise an isotropic energy source using measurements from distributed sensors, a practical algorithm, localising via averaging[2], is considered. It is a simple algorithm that is used for calculating the position instantly when the data is received from each and individual sensor.

The proposed model takes the known location of the sensor as weight and multiplied with the received signal strength (RSS) to pinpoint the source location for all the sensors, i.e.

$$\bar{x}_S = \frac{\sum_{j=1}^{n} \bar{x}_j \cdot g(s)}{\sum_{j=1}^{n} g(s)} \tag{2.3.1}$$

where $\bar{x}_S$ is the source is location and g(s) is the RSS that dictates how greatly a contribution different sensor make in computation of *Eq. 2.3.1* based on their received signal strength (the estimated distance from the source) [2]. The *Eq. 2.3.1* also dictates that the more sensors there are contributing to the localisation, the more accurate is the result.

2.3.1 Source localising example

Example, in our case, we only want to pinpoint the source on a two dimensional area looking from above. Figure 2.3.1.1 shows a placement of three nodes with an equal distance relative to each other.

When a signal is received of strength $\boldsymbol{x_j}$ by each $\boldsymbol{j}$th sensor, the coordinates by the averaging algorithm will result to:

$$x = \frac{x_a \cdot 500 + x_b \cdot 100 + x_c \cdot 900}{x_a + x_b + x_c} \tag{2.3.1.1}$$

$$y = \frac{x_a \cdot 900 + x_b \cdot 100 + x_c \cdot 100}{x_a + x_b + x_c} \tag{2.3.1.2}$$

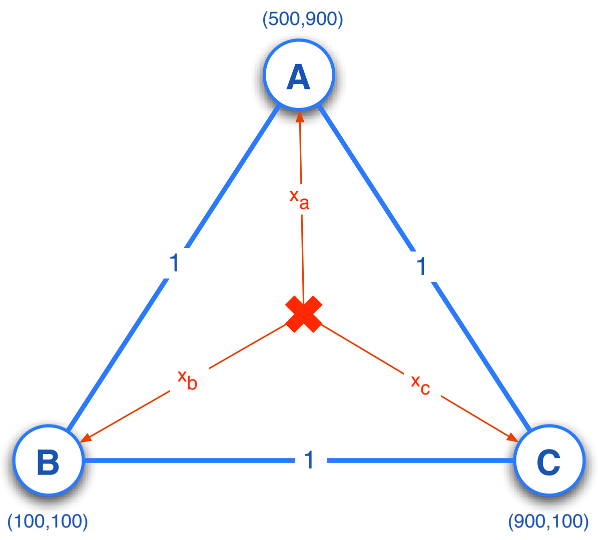

Figure 2.3.1.1 - Source localisation example with three nodes.

And if, for the ideal case, now considered that the signal strength received by each node is the same, $x_a = x_b = x_c$, the following coordinates will be given:

$$x = \frac{x_a(500+100+900)}{3 \cdot x_a} = 500. \tag{2.3.1.3}$$

$$y = \frac{x_a(900+100+100)}{3 \cdot x_a} \approx 367. \tag{2.3.1.4}$$

This will give the coordinates in two dimensions of the source to be located at (500,367).

3

Realisation

The implementation of the complete work have been mainly focused on TDMA with the E-ASAP protocol. The attempt has been made to realise it in as simple manner as possible. Consequently, the implementation of application is designed to be general for all nodes. In this way, the simplicity can be maintained in design and the system more reliable. However, one node, the base node, had to be implemented a bit differently due to its serial communication with the computer. For having a sensor on a field with the main purpose of communicating wirelessly has no need for serial communication. Due to this reason, the implementation of the serial communication is removed for sensors in general. This is done easily by a simple adjustment made during the installation procedure of the node. Although, the base node itself acts like a normal sensor in the network with the extra ability of serial communication.

In the coming sub chapters, it is explained how each part of the problem, addressed in theory chapters, are implemented and how easily the TDMA with the E-ASAP protocol can be used for any application. Then in the final stage, it is shown how the TDMA/E-ASAP is used for tracking a light source.

3.1 Time Division Multiple Access (TDMA)

The importance of a good real-time behaviour of the TDMA lies on its accuracy. Each slot time, ***interval***, has to be as accurate as possible because the whole network depend on this timings. For this, TinyOS/nesC provides a good timer for this purpose and is naturally used. Another importance relies on the *time* variable that is calculated internally, Figure 3.1.1.

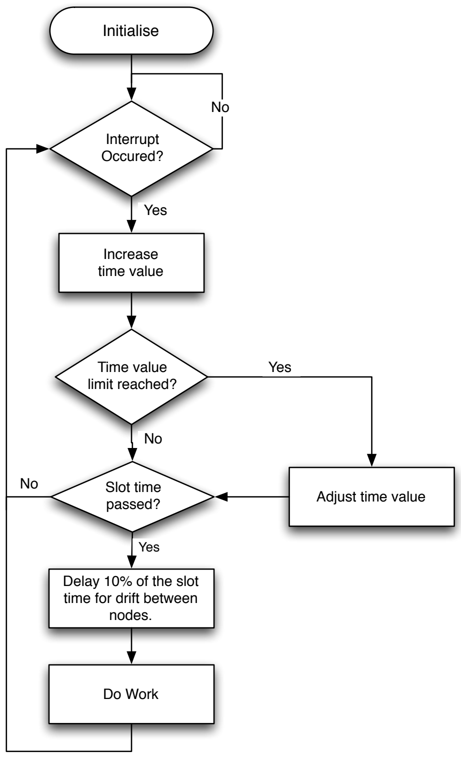

Figure 3.1.1 - Time slot occurrence flow chart

The timer's interrupt is used for determining which time slot the system is currently in. A time slot is predefined value and is determined by

```
currentSlot = (time / interval) % frame.
```
(3.1.1)

where ***interval*** is the length of the slot and***frame*** is the TDMA period. However, a safety net for the overflow is required for the *time* variable. When passed a predefined limit, it is adjusted by the following algorithm:

```
time = currentSlot * interval + time % interval
```
(3.1.2)

It is reset and increased to a value, with its remainder, to the minimum value corresponding to the slot number.

For example, if the following variables are currently set in the system:

Table 3.1.1 - Time overflow example

Variable	Value
interval	300
currentSlot	2
frame	4
time	50010
LIMIT	50000

Here, the current state of time is at 70 % in slot 2 and counting. This state has to be exactly the same after the *time* variable is adjusted. By the values given above and using Eq. 3.1.2, *time* variable will result to:

```
time = 2 * 300 + 50010 % 300

time = 600 + 210 = 810.
```

By again using the equation (3.1.2), the current state of the system is still 70% in slot 2 and therefore no changes has contributed to the TDMA after the adjustment.

3.2 Extended Adaptive Slot Assignment Protocol (E-ASAP)

Initially, with the setting of the the TDMA period to four and slot zero, the new node starts collecting INF packets from its neighbourhood for some time, in our case five periods, before joining the network, Figure 3.2.1.

When collecting the information about the contention area, the new node needs to have the knowledge of all the neighbours before adding the information about the hidden nodes. That's why there is a delay in updating hidden nodes, currently set to two TDMA periods. Without this delay, a neighbour will be added as a hidden node although it can be a neighbour. However, the system will adapt in time because of the release node procedure, described in chapter 3.2.1.2, but this delay is unnecessary and a waste of resource.

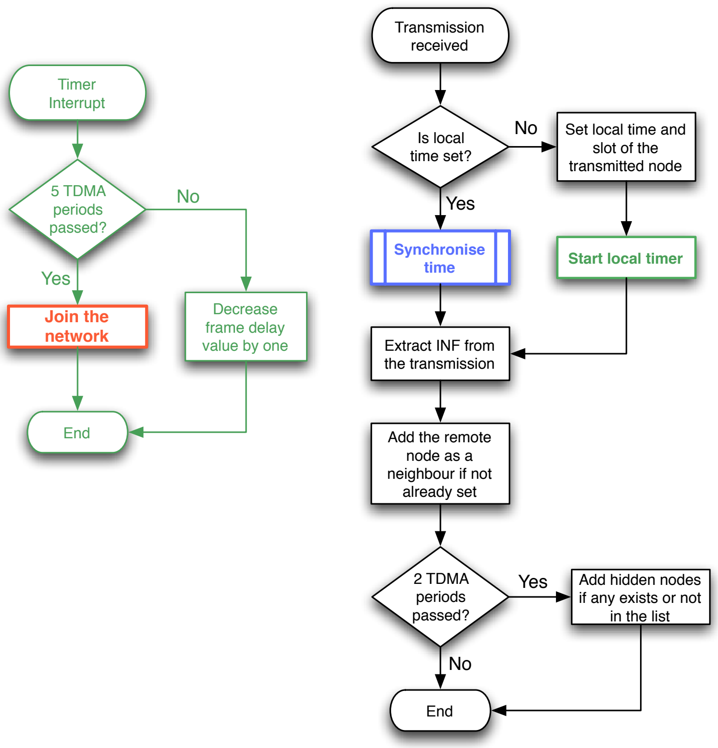
Timer Interrupt
5 TDMA periods passed?
No
Yes
Join the network
Decrease frame delay value by one
End
Transmission received
Is local time set?
No
Set local time and slot of the transmitted node
Yes
Synchronise time
Start local timer
Extract INF from the transmission
Add the remote node as a neighbour if not already set
2 TDMA periods passed?
Yes
Add hidden nodes if any exists or not in the list
No
End

Figure 3.2.1 - Procedure for joining the network for a new node.

3.2.1 Slot assignment

When all the information is gathered from the neighbourhood about the contention area, the new node is safe to join the network. The procedure is split into two parts. First, the node searches for the first unassigned slot, and sets it to itself. Then it checks if it has doubled its TDMA period during the procedure and sets the neighbourhood information accordingly to the new period. The procedure is shown in Figure 3.2.1.1.1.

Once joined, each nodes in the contention area is monitored about its presence by a variable that it is connected to, denoted ***ackAlive***. For each transmission received by the neighbouring node, ***ackAlive*** is increased by one for the transmitting node and respectively for all the hidden nodes that are in the INF packet. Correspondingly, the variable is decreased by one at slot zero for each TDMA period. By this procedure, the node is considered alive in the network as long as it transmits.

In our case, When five TDMA periods have passed, the new node joins the network and will have its ***ackAlive*** value set to sum of all transmission received during this delay. Ideally it will be set to five when joining the network.

3.2.1.1 Getting an unassigned slot

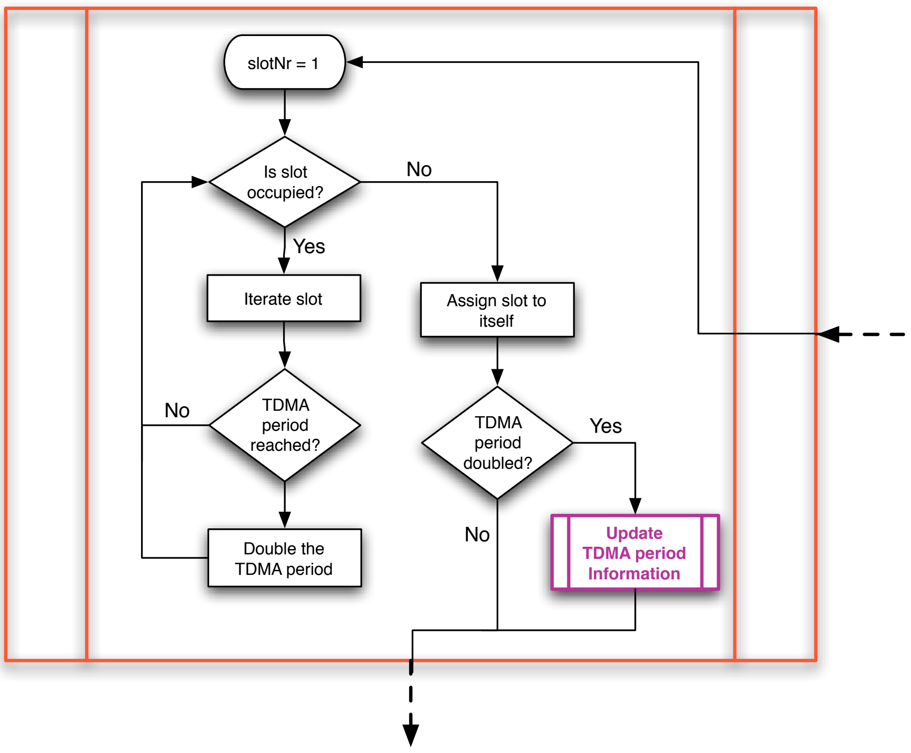

Figure 3.2.1.1.1 - Slot Assignment procedure.

When searching for an unassigned slot, all the slots that are occupied by the nodes in the contention area are ignored till the first unassigned slot is found and is set to itself by the new node. If the TDMA period has increased during the search procedure, doubling of TDMA period is processed with the information about the contention area must that is preserved. This is handled by the Doubling TDMA period routine, described in chapter 3.2.1.3.

3.2.1.2 Releasing slot assignment

When a node leaves the network, there will be no transmission received by the neighbouring nodes. Consequently, the variable ***ackAlive*** will decrease by one on slot zero for each TDMA period and eventually will reach zero. It is then considered that the node have left the network. All information containing the exit node will be then removed and the Halving TDMA period routine, described in chapter 3.2.1.4, is applied for the maximum utilisation of the network. This is valid for all the nodes in the contention area because the information on the hidden nodes is also in the INF packet of the neighbours. If only one neighbour has the information on a certain hidden node, when exited, the hidden node will also be considered to have left the network.

3.2.1.3 Doubling TDMA period routine

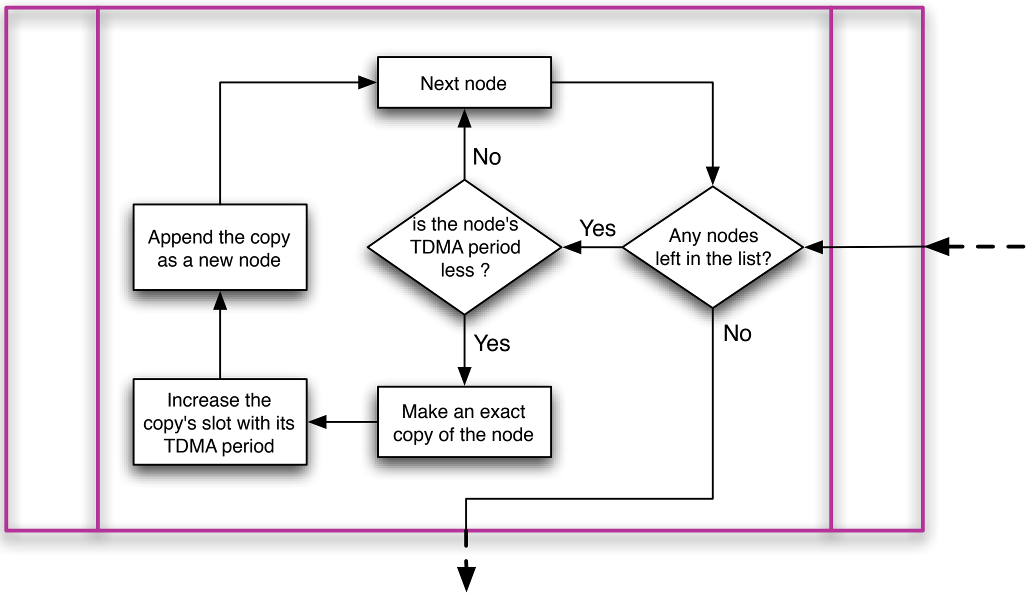
Next node
No
is the node's TDMA period less ?
Yes
Any nodes left in the list?
Append the copy as a new node
Yes
No
Increase the copy's slot with its TDMA period
Make an exact copy of the node

Figure - 3.2.1.3.1 - Doubling TDMA period procedure.

When doubling the TDMA period, for all the nodes in the contention area, their information is copied and added as a new nodes with their slot added with their TDMA period. This is to provide collision free communication when a new node tries to join the network as described in the example at the end of the chapter 2.1.2.1.

3.2.1.4 Halving TDMA period routine

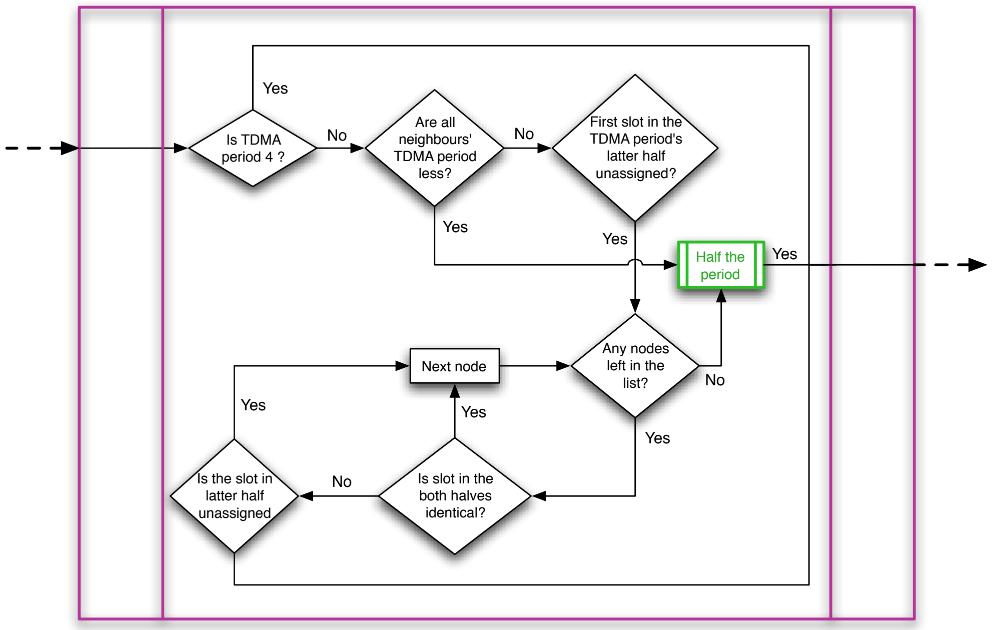
Yes
Is TDMA period 4 ?
No
Are all neighbours' TDMA period less?
No
First slot in the TDMA period's latter half unassigned?
Yes
Yes
Half the period
Yes
Next node
Any nodes left in the list?
No
Yes
Yes
Yes
Is the slot in latter half unassigned
No
Is slot in the both halves identical?

Figure 3.2.1.4.1 - Halving TDMA period procedure

Generally, the TDMA period is halved when the period's slot information is identical on both halves or no nodes are assigned to the latter half. But when requested on the minimum period length, set to four by the E-ASAP, it is discarded.

3.3 Time Synchronisation

Time synchronisation is implemented as described in the chapter 2.2. Even so, as generally in computing time, overflow must be handled correctly or the whole network is most likely to collapse.

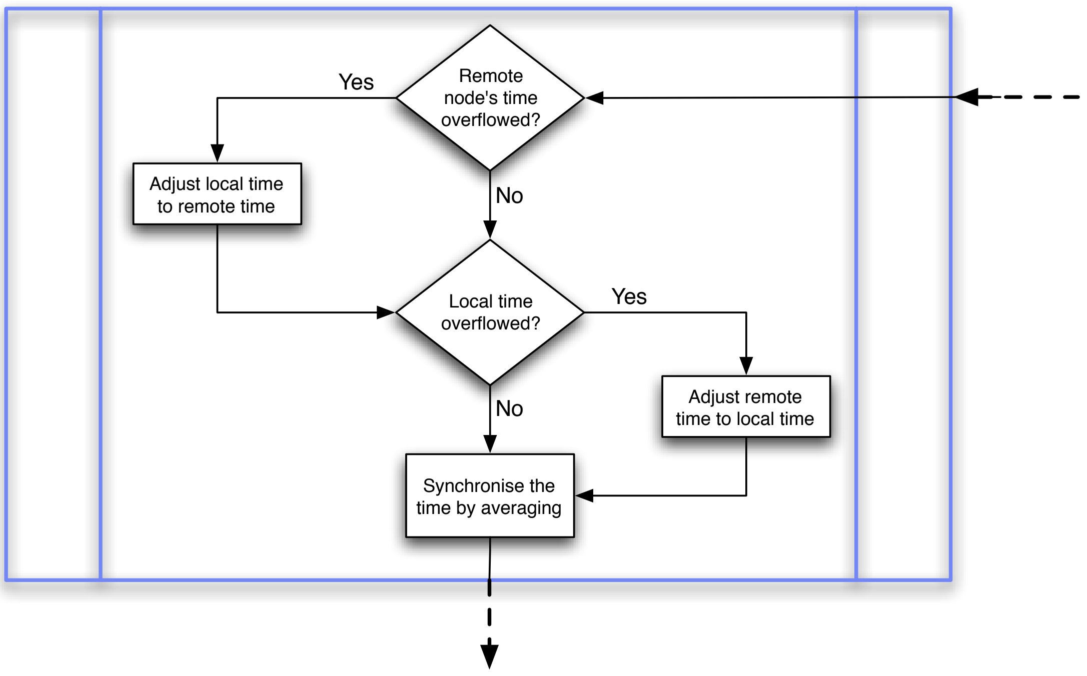
Remote node's time overflowed?
Yes
No
Adjust local time to remote time
Local time overflowed?
Yes
No
Adjust remote time to local time
Synchronise the time by averaging

Figure 3.3.1 - Time synchronisation procedure

When the first transmission is received, the local time is set with the same value as the transmitting node and starts counting. Once done, the time value is then synchronised for every received transmission as shown in Figure 3.3.1. The danger lies in when one of the nodes, either local or remote, has its local time value overflowed. These time values must be checked and adjusted if necessary before the time is synchronised. This is done as following:

If the remote node's time has overflowed, the local time is adjusted as:

```
time = currentSlot * interval + time % interval          (3.3.1)
```

If the local node's time has overflowed, the remote node's time is adjusted:

```
rTime = currentSlot * interval + rTime % interval
```
(3.3.2)

And finally, the time is synchronised as:

```
time = (time + rTime) / 2
```
(3.3.3)

3.4 Component TDMA

So far, all the implementation has been focused on TDMA and E-ASAP. Naturally, this was the first attempt to realise the ad hoc network by implementing the mentioned protocols. Also naturally, it has been developed as its own component called ***TDMA*** with an interface for sending and retrieving data, see appendix A-1 for the Application Programming Interface (API).

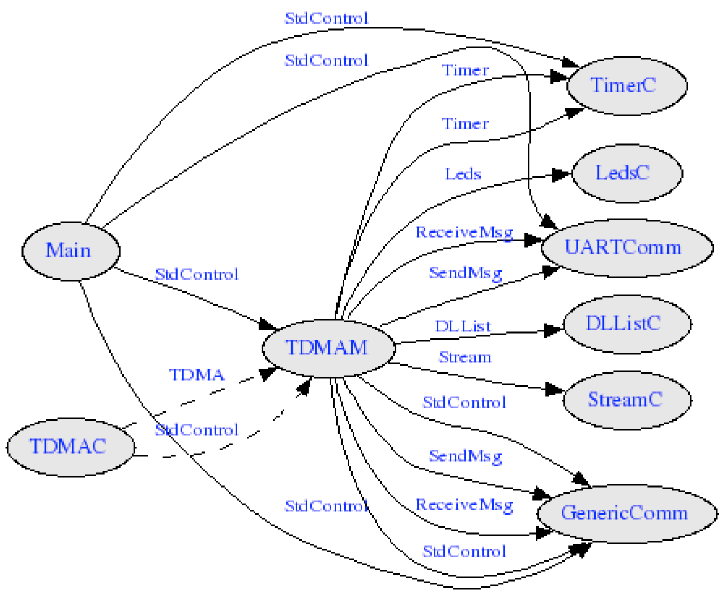

Figure 3.4.1 - TDMA component "wiring" layout.

There are additionally two more components developed as libraries for to be used by the TDMA. The first library is ***DLList*** and is used for manipulating the INF information that is held by the node. The second library is ***Stream*** and is used for packaging and extracting the the INF data to and from the raw byte format that is used for transmission.

The TDMA component can be applied in any end user application, whether it is a source localisation or security surveillance.

3.5 Component ObjectTrack

For the purpose of localising a source, in our case light, a new component called ***ObjectTrack*** is developed. It has been wired with the TDMA component for the transmission of data throughout the network. The ObjectTrack component does not need to know or care how the data is passed in the network. All it does is measuring the sensor data and hands over the value to the TDMA component by its interface, described in appendix A-1. It lets the TDMA component worry about how and when the data needs to be transmitted within the network.

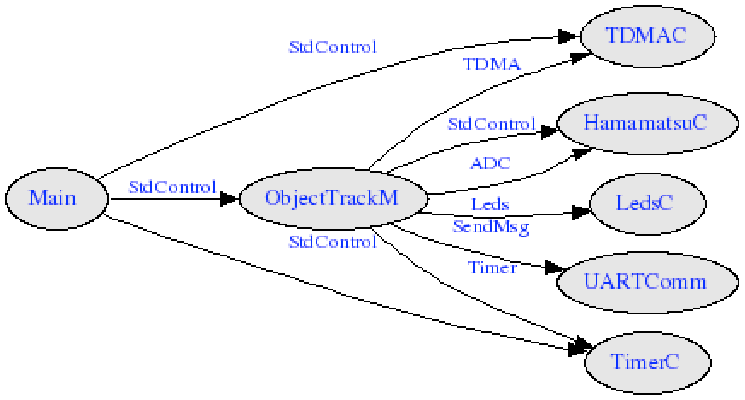

Figure 3.5.1 - ObjectTrack component "wiring" layout

In ObjectTrack component, all nodes in the network are set in passing mode were they simply send the data to the base station. The base station itself acts only as a receiver and passes the data by using the serial communication to the computer. The serial data at the computer end is captured with the structure shown in Figure 3.5.2.

7E	TYPE	NN	RESERVED	DEST. ADDR	MSG TYPE	GRP ID	DATA	CRC	7E
1B	1B	1B	5B	2B	1B	1B	NN B	1-2 B	1B

Figure 3.5.2 - Serial data format sent to the computer.

The leading and trailing framing byte, **0x7E**, defined by TinyOS, marks the complete packet of the data that is sent. The third byte contains the length of the data (in bytes) that this packet is holding. And finally the data itself starts from the byte 12. In our case, the encapsulated data structure is shown in Figure 3.5.3.

Time Stamp	Node ID	Slot	Frame	Sensor Data
2B	2B	1B	1B	2B

Figure - 3.5.3 Encapsulated data format within the serial data packet.

The serial line is monitored periodically by the Graphical User Interface and displays the results conveniently.

3.6 Graphical User Interface Application

A simple Graphical User Interface, GUI, application has been developed using KDE, Qt and QWT libraries under Linux operating system.

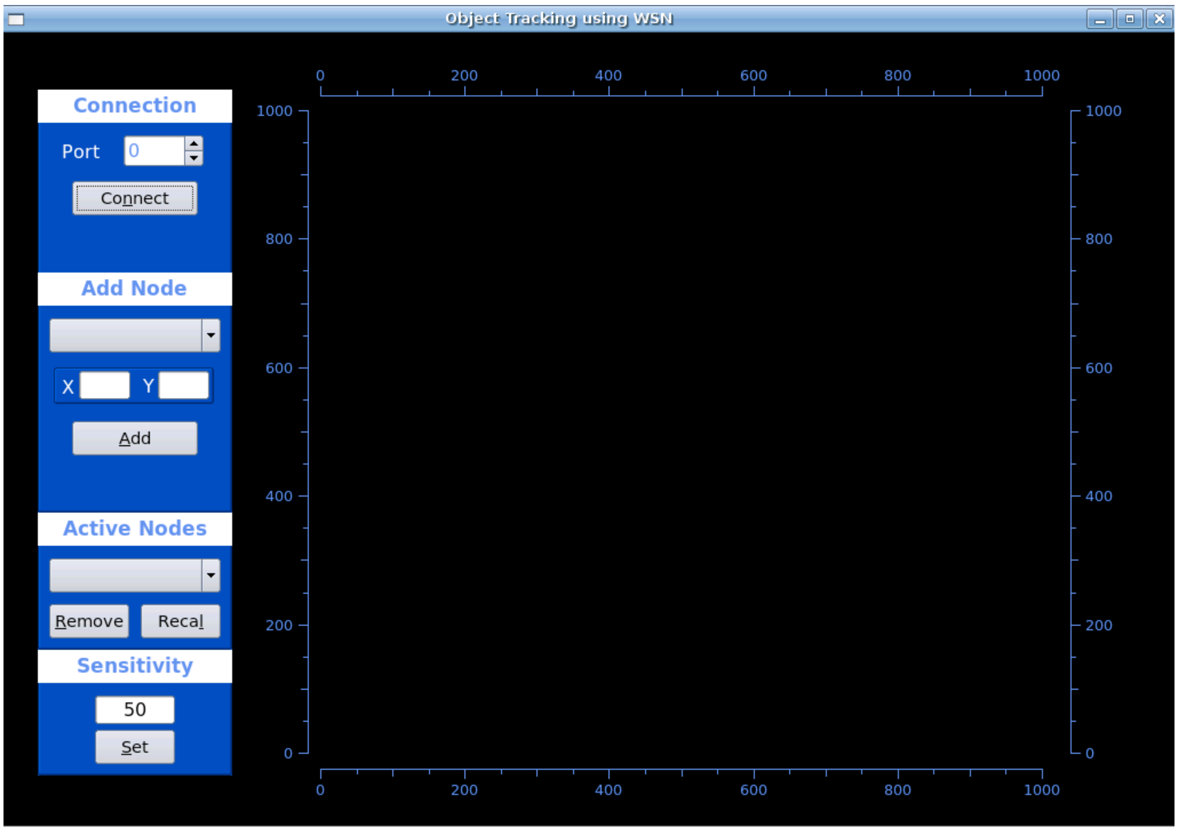

Figure 3.6.1 - GUI application for tracking object

The application first needs to connect to the base station, where after, on success, it monitors the serial line for data transmissions from the node. However, in the first stage, all the data is ignored except the nodes' ID. For each new node, the node is added to the *ComboBox* box in the frame of *Add node*. From this frame, a node can be added to their hypothetical coordinates. For example, if four nodes are placed in a 'diamond' configuration, Figure 3.6.2,

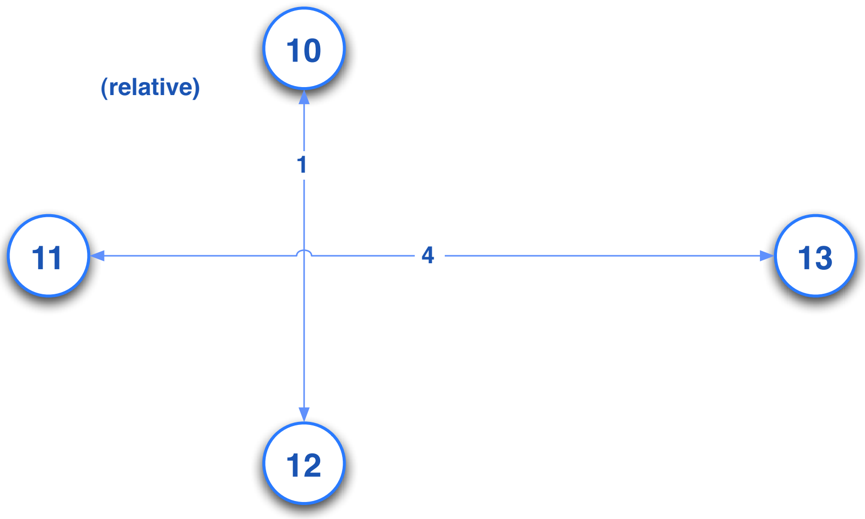

Figure 3.6.2 - Diamond shape configuration example for placing on the application.

they may be represented in the application with the coordinates **10** = (300,700), **11** = (100, 500), **12** = (300, 300) and D = (900, 500), as shown Figure 3.6.3.

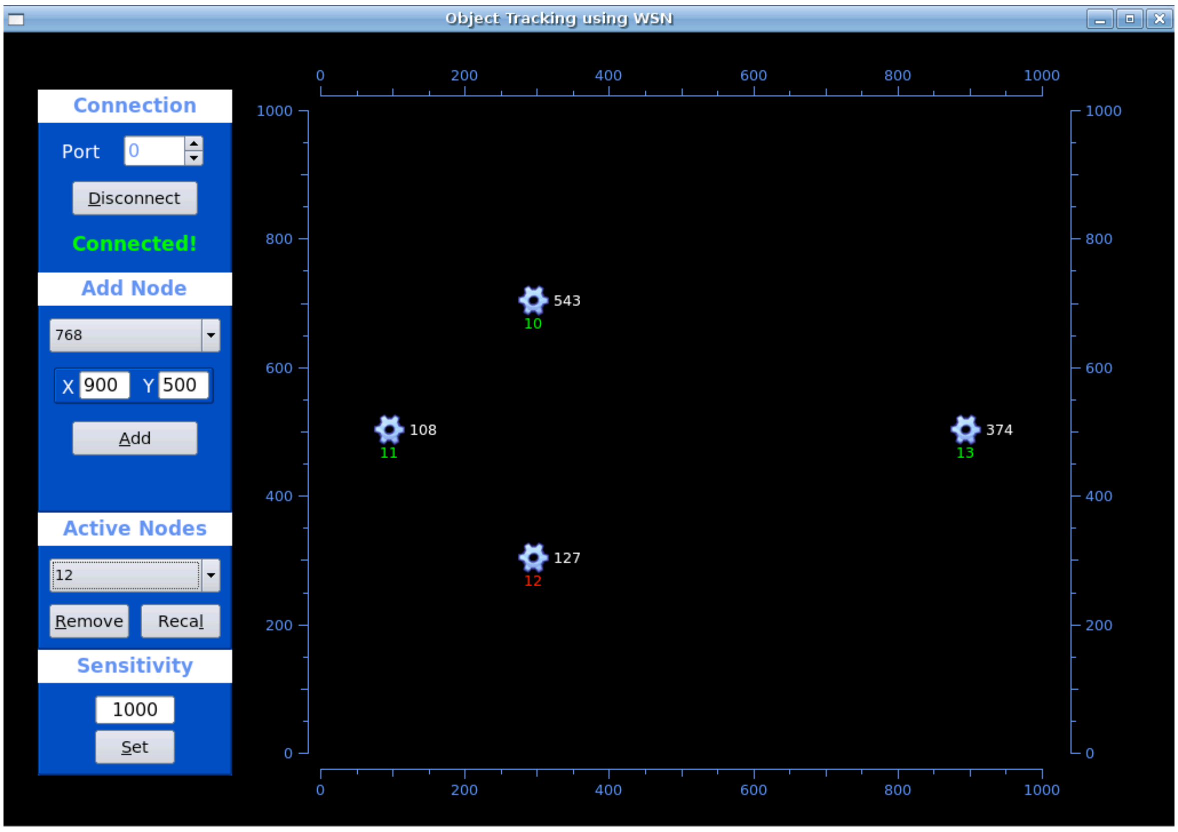

Figure 3.6.3 - GUI with the diamond layout of the sensors

Once placed, the nodes ID's are added to the *ComboBox* in the frame of *Active nodes*. On the *field* layout, the node's ID will be marked red until it is calibrated with the background light for five TDMA periods. Once calibrated, the node's ID label will change its colour to green and its current sensor value displayed by the right hand side of the image. If a nodes needs to be replaced, it can be removed and added again. A node can also be recalibrated by selecting the node's ID from the *ComboBox* in the frame of *Active nodes* and by pushing the ***Recal*** button. Finally, at the bottom, sensitivity value can be chosen. This value applies for minimum change in reading by each sensor required for assuming a source is present.

If a source is placed between the nodes ***10***, ***11*** and ***12***, the application will present the result as shown in Figure 3.6.4.

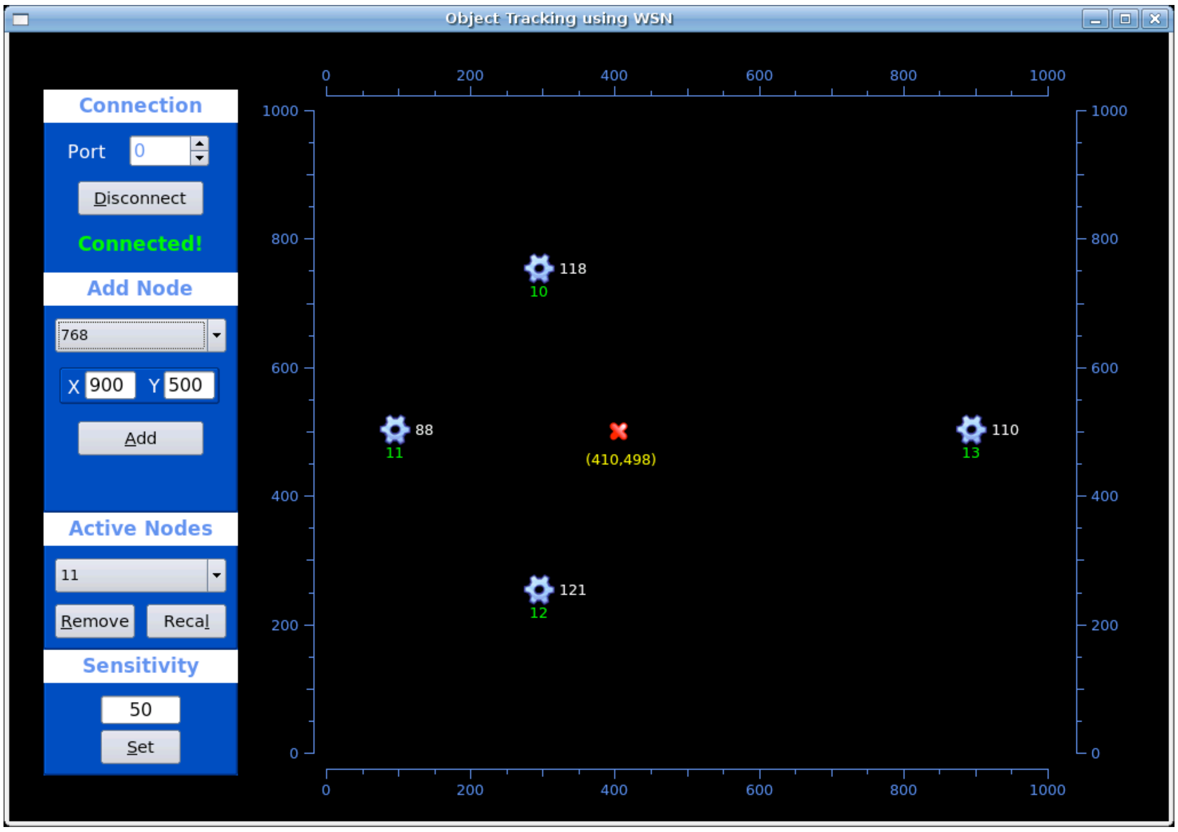

Figure 3.6.4 - Example of the result presented by the GUI application when a source is placed between nodes 10, 11 and 12.

4 Results

The test are split into three phases for covering the behaviour of the complete system.

In the first phase, the functionality of the TDMA and E-ASAP protocols are determined. Here, two different scenarios are set to show how the system responds to the dynamical change in the network. In the first, only the neighbourhood's functionality is determined. Later, the response of the system is determined where dynamic changes are in the whole contention area.

In the second phase, time synchronisation is evaluated for its accuracy. Two different tests are conducted where, in both, a node is deliberately been injected with a time error. There after, the system response is monitored on how it adapts to the injected error.

In the final phase, source localisation is evaluated on a test field. On this field, the sensors are placed with different configuration and then presented on the GUI application with their result.

4.1 TDMA and E-ASAP

For testing the complete dynamics of the network, the tests are divided into two different categories. First, a simpler test is conducted where only the dynamical functionality of the neighbourhood is evaluated where nodes are entered and exited randomly. Then the test is finalised with the complete network, i.e. with contention area. On both tests, six nodes have been used with one node, node ***10***, as the base station that is connected to the computer using the serial communication. On both tests, node ***10*** initialises the network and is set with slot ***one*** with the TDMA period ***four***. The time slot is set to 300 milliseconds.

The data retrieved from the base station comes in the format as shown in Figure 4.1.1.

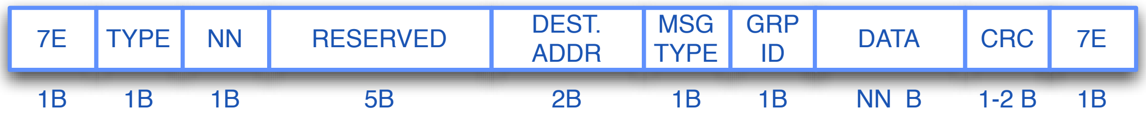

Figure 4.1.1 - Serial data format sent by the base station.

The data stream is encapsulated with the *framing byte*, 0x7E, and the third byte contains the information on how many data bytes there are in the data field. The data field itself is dynamic depending on how many nodes there are in the network, whether neighbour or hidden, seen from the base station's point of view. Figure 4.1.2 shows the structure of the encapsulated data.

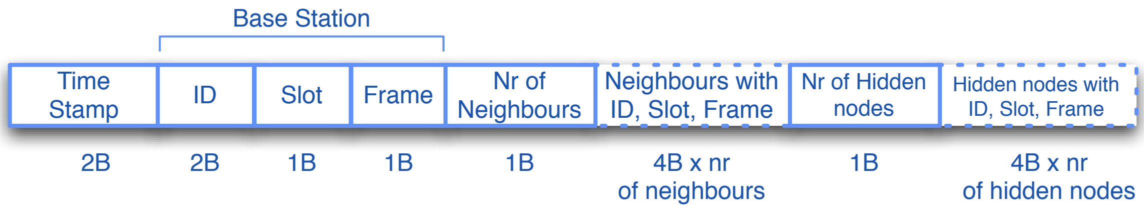

Figure 4.1.2 - Encapsulated data within the serial data packet.

4.1.1 Dynamics of the network with neighbouring nodes.

For testing the dynamics of the neighbourhood, the configuration is simple as shown in Figure 4.1.1.1. The network is initialised,as mentioned, with node ***10*** and where after nodes enters and leaves the network dynamically. The result is shown in Table 4.1.1.1.

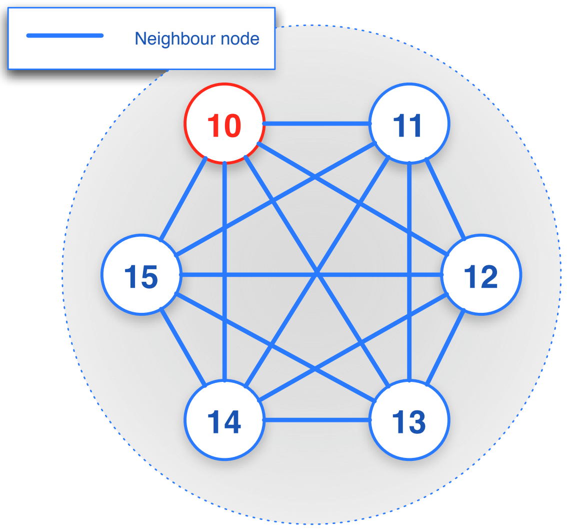

Figure 4.1.1.1 - Simple configuration for the test of dynamics in neighbourhood.

Table 4.1.1.1 - Network test result on neighbourhood

	NODE ID					
Action	10 slot / frame	11 slot / frame	12 slot / frame	13 slot / frame	14 slot / frame	15 slot / frame
Initialised with 10	1 / 4	-	-	-	-	-
ADDED 11	1 / 4	2 /4	-			
ADDED 12	1 / 4	2 / 4	3 / 4	-	-	-
REMOVED 11	1 / 4	-	3 / 4	-	-	-
ADDED 13	1 / 4	-	3 / 4	2 / 4	-	-
ADDED 11	1 / 8	4 / 8	3 / 8	2 / 8	-	-
ADDED 15	1 / 8	4 / 8	3 / 8	2 / 8	-	5 / 8
REMOVED 13	1 / 8	4 / 8	3 / 8	*	-	5 / 8
ADDED 14	1 / 8	4 / 8	3 / 8	-	2 / 8	5 / 8
ADDED 13	1 / 8	4 / 8	3 / 8	6 / 8	2 / 8	5 / 8
REMOVED 11	1 / 8	-	3 / 8	6 / 8	2 / 8	5 / 8
REMOVED 15	1 / 8	-	3 / 8	6 / 8	2 / 8	-
REMOVED 13	1 / 4	-	3 / 4	-	2 / 4	-
REMOVED 14	1 / 4	-	3 / 4	-	-	-
REMOVED 12	1 / 4	-	-	-	-	-

The results were very satisfying. The respond of the system is as it is defined by the requirements set in the TDMA/E-ASAP. However, there is one remark on the result that needs to be commented. When node ***13*** is removed from the network for the first time, (*), it became as a hidden node for a couple of TDMA periods before it disappeared completely. This happens when one node has been removed from all the neighbouring node's list except at least one, i.e. the corresponding ***ackAlive*** variable is still not zero. The node that still have the removed node on its list will send its INF packet containing that the is still in the neighbourhood. This results to a hidden node for the other neighbouring nodes that already have removed it. This is not a faulty behaviour. A dynamic network cannot assume when a node should enter or leave.

4.1.2 Dynamics of the network with hidden nodes.

For testing the hidden nodes dynamics, the configuration of the nodes are split into two zones, Figure 4.1.2.1, where node ***12*** acts as a bridge between these zones. Once again, the network is initialised with node ***10*** where after nodes enters and leaves the network dynamically. The result is present in Table 4.1.2.1.

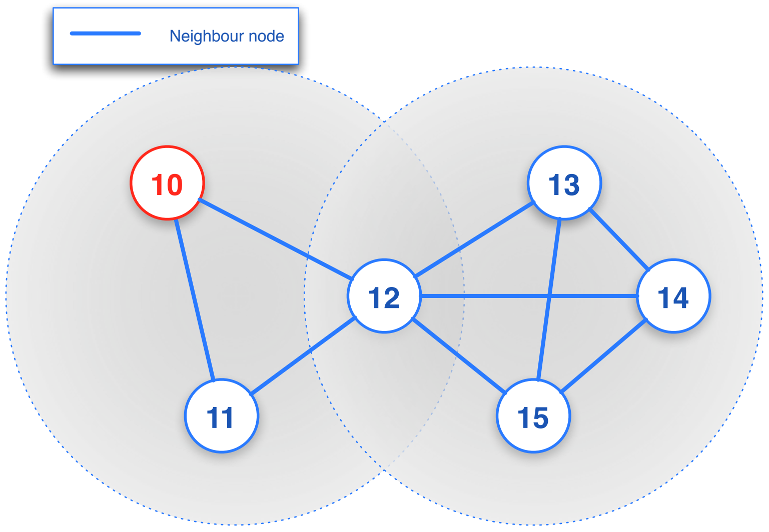

Figure 4.1.2.1 - Configuration for the test of dynamics with hidden nodes.

Again the results were satisfying. Here, not only the hidden nodes dynamics can be seen but also the consistency of the network when doubling the TDMA period requires repeating the information of the nodes. When node is ***11*** is added to the network, row 4, nodes ***10*** and ***12*** must change their TDMA period to ***eight*** to be able to fit all nodes in the period. Node ***14*** does not need to increase its period and therefore is repeated in the period of ***eight*** instead by its neighbours. From the node ***14***'s point of view, it is only sending its transmission on slot ***three***.

Table 4.1.2.1 - Network test result on hidden nodes

	NODE ID					
Action	10 slot / frame	11 slot / frame	12 slot / frame	13 slot / frame	14 slot / frame	15 slot / frame
Initialised with 10	1 / 4	-	-	-	-	-
ADDED 12	-	-	2 / 4	-	-	-
ADDED 14	1 / 4		2 / 4	-	3 / 4	-
ADDED 11	1 / 8	4 / 8	2 / 8	-	3 / 4 7 / 4	-
ADDED 13	1 / 8	4 / 8	2 / 8	5 / 8	3 / 8	-
ADDED 15	1 / 8	4 / 8	2 / 8	5 / 8	3 / 8	6 / 8
REMOVED 14	1 / 8	4 / 8	2 / 8	5 / 8	-	6 / 8
REMOVED 11	1 / 4	-	2 / 8	5 / 8	-	6 / 8
REMOVED 13	1 / 4	-	2 / 8	-	-	6 / 8
REMOVED 15	1 / 4	-	2 / 4	-	-	-
REMOVED 12	1 / 4	-	-	-	-	-

4.2 Time synchronisation

For testing the time synchronisation, only one computer can be used. Internal timer of the nodes cannot be trusted due to their own independent drift and therefore an external timer is used for determining the time difference when each node transmit their INF package. Also, due to the limitation on the physical layer of serial communications on the computer, only two nodes could be connected and tested at the same time.

Two tests are conducted for determining the accuracy of time synchronisation. For the first test only two nodes are present in the neighbourhood. In the second test, six nodes are included in the network as a whole. And on both tests, a transmission to the computer is sent on slot ***zero*** by both nodes that are connected to the computer. The time difference on their time arrival of transmission is then monitored. Slot time, ***interval***, is set to 1000 milliseconds with an injected time error of 500 milliseconds. Unfortunately, the external timer's interrupt interval could not be set lower than four milliseconds on the computer. Consequently, this limits the accuracy of the test with the minimum measurable error margin of four milliseconds.

4.2.1 Time synchronisation test with two nodes

On the first test, the network is initialised with node ***10*** where upon after some time, node ***11*** is entered. When the first transmission is received by node ***11*** from node ***10***, the local time is set to node ***10***'s time added with the time error of 500 milliseconds. Then the time difference is measured simultaneously that is sent by *their* slot ***zero***. The result shown in Figure 4.2.1.1.

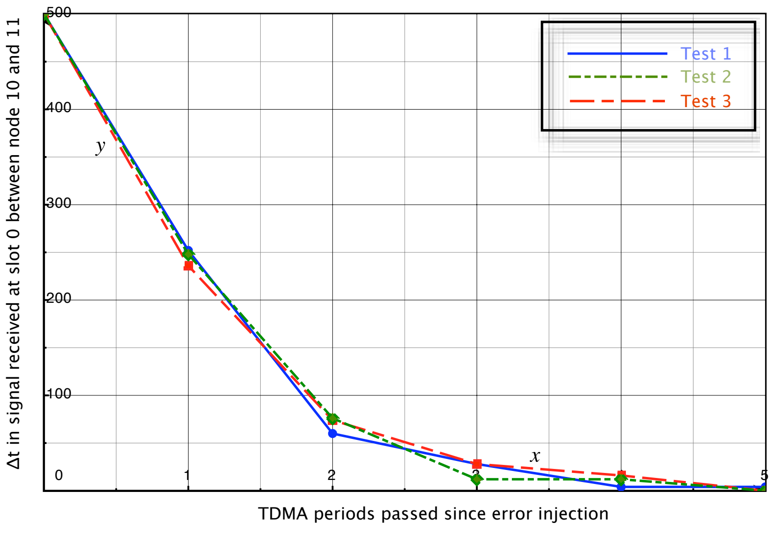

Figure 4.2.1.1 - Synchronisation test with injection error of 500 milliseconds with two nodes.

As it can be seen, within five TDMA periods, the error is corrected as expected from the simulation conducted in chapter 2.2.1.

4.2.2 Time synchronisation test with six nodes

On the final test of the synchronisation, the network is initialised with node ***10*** where upon nodes **12**, **13**, **14** and **15** are entered. When all the nodes in the network are aware of each other, node ***11*** is introduced to the network with its fixed slot at ***six***. Upon first transmission received by node ***11***, the local time is set to transmitted node's time added with the time error of 500 milliseconds. Then the time difference is measured simultaneously that is sent by node ***10*** and ***11*** at *their* slot ***zero***. The result shown in Figure. 4.2.2.1.

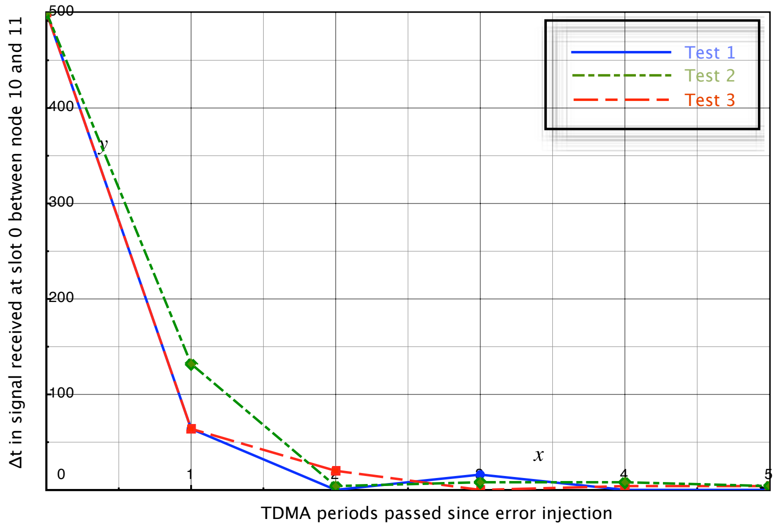

Figure 4.2.2.1 - Synchronisation test with injection error of 500 milliseconds with six nodes.

As expected from the simulation, the correction is improved when more nodes are in the neighbourhood. However, the error is corrected within ***four*** instead of ***two*** TDMA periods which is twice as long as the simulation shows. There is no way of telling why there is a delay on the result compared to the simulation. But in our case, we are only interested that the correction is made more quickly when more nodes are involved in the synchronisation method, which it does as this test shows.

There is a constant error always present in the network, which was expected, around four milliseconds. This error is present due to the limitation of the external time interval set by the computer as mentioned previously. Nevertheless, there is delay present due to the communication delay between the nodes. This delay is not considered in the simple time synchronisation algorithm that has been used. However, for the scope of this work, this margin is acceptable.

4.3 Localising the light source

All localising tests are done on a field 1000 x 1000 mm, Figure 4.3.1, with a simple table lamp from IKEA.

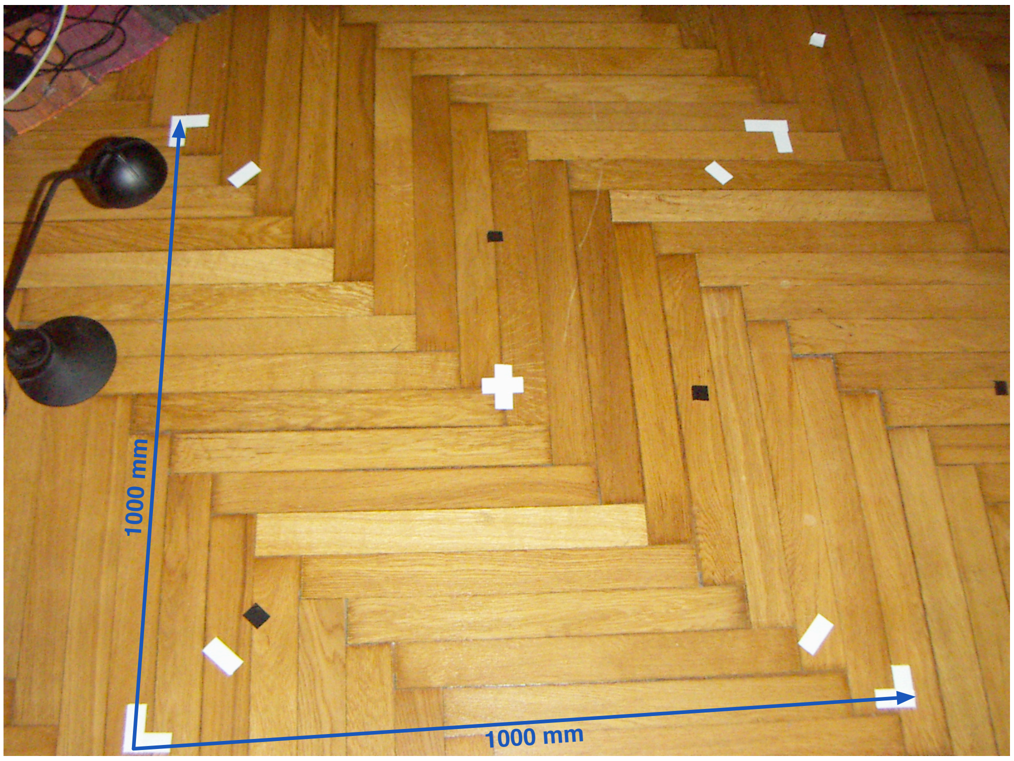

Figure 4.3 Test field for the source localisation.

Five nodes are placed in different configuration to test the capability of the network and the GUI application for determining the location of the light source. One node, the base node, acts like a passive receiver and channels the data further on to the computer via its serial communication. When locating the source, localising via averaging[2] algorithm is used to show how it is effected by the different configurations.

4.3.1 Configuration X

First a simple symmetric configuration is set for testing, Figure 4.3.1.1. This is assumed to be most applicable configuration where the object is triangulated to a fix position with at least three sensors. Once the sensors are placed and have been calibrated to the background light, the lamp is placed on four different positions and the results are presented, Figure 4.3.1.1.

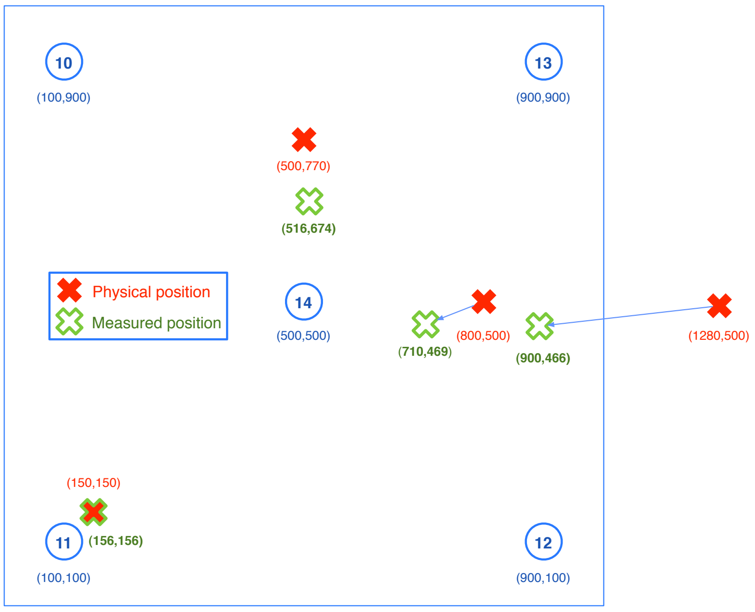

Figure 4.3.1.1 - Symmetric layout configuration for optimal source location.

The results were somewhat a surprise. The measured position, for the most part, was a bit too distant than the physical location. There are mainly two contributions to this error. The first one is contributed by considering if the actual source is placed where it is coordinated to the corresponding layout. This is however a minor error. The second contribution, far beyond more important, is that the sensors are not calibrated to the source before the measurement. This gives different sensor readings for the same radiation. This is important to considerate because the position is calculated linearly by their values although the radiation is reduced by the distance squared. The last comment that is important to point out is that a source is always considered, in this system, to be within the area of the nodes. When the source was put outside the test area, it showed that source is in the edge but inside the area.

4.3.2 Configuration T

In this configuration, an attempted is made to show that if the sensors are lined up on a straight line relative to each other, a distant sensor from the source will “pull” the result towards itself. Once again the sensors are placed and then calibrated to the background light before the lamp is placed on four different positions. The results presented on Figure 4.3.2.1.

The results were as excepted, the measured positions is “pulled” towards the alignment of the sensors placements’ relative line. When using the localising via averaging As a result, this is not a recommended configuration.

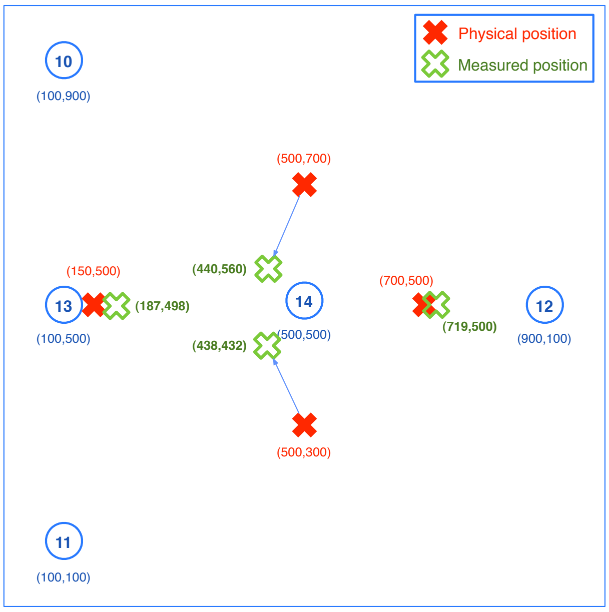

Figure 4.3.2.1 - Lined configuration

4.3.3 Configuration Y

Similar to the T configuration as described in chapter 4.3.2, a small modification have been made to show even if there are nodes triangularly placed, the sensors in line will still influence the measurement by “pulling” the result to itself.

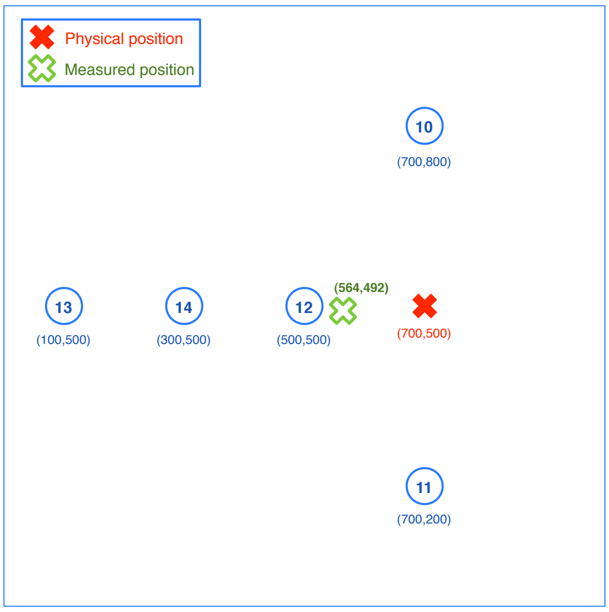

Figure 4.3.3.1 - Y configuration.

As expected, the outcome of the result is acknowledged, Figure 4.3.3.1. The measured position is “pulled” towards the left simply because of the weight by the nodes that are in a line relative to each other. This is also not a recommended configuration.

4.4 Test summary

The resulting measurements conducted on TDMA/E-ASAP and time synchronisation has been satisfactory. The error were small if not insignificant and has shown by using the TDMA/E-ASAP protocol with time synchronisation, it is very suitable for wireless sensor networks.

Considering source localisation, first of all it is fair to mention that the measurements has given poor accuracy. This was mainly a result of lack of calibration to the light source itself. The sensors should have been individually calibrated first to the environment then the with the source before the test were conducted. Having said that, it has been shown that the method localising via averaging gives a position that is extremely dependent on the configuration of the sensor layout on the field. It has shown that the sensors should not be placed in such way that they are aligned to a line relative to each other. It is also important to point out that each measured value is averaged to their reading although the strength of the source radiation is reduced by the distance squared. It is recommended to enhance this algorithm slightly by taken the square root of the measured value before averaging. And finally, the quantity of the nodes should also be considered. More nodes leads to a better accuracy.

5 Conclusion

Time Division Multiple Access (TDMA) with the protocol Extended Adaptive Slot Assignment (E-ASAP) is a challenging subject, where a simple solution is hard to implement on Wireless Sensor Networks WSNs. The performance of the system when using the mentioned protocols have been satisfactory but more tests are required to fully show the complete behaviour. Another issue lies in data handling. So far, the data is broadcasted by each node in the network. This is not acceptable in a larger and more complex network and therefor it is suggested to be improved with multi hoping capability as mentioned in the next chapter, Future work. Although with its limitations, it is shown that it works very well.

Time synchronisation was not really a part of the thesis but needed to be included for its importance. However, the tests were restricted by the limitation set by the computer and needs to be additionally expanded for confirming the complete performance. Even though the performance has shown to be satisfactory.

Finally, the source localisation was a bit of a disappointment. The error between the measured and physical positions were to large. This was not a failure of the system but mostly contributed by the lack of calibration to the source. This can be adjusted easily. Another consideration that needs to be addressed is the algorithm for calculating the location needs to be improved slightly. At the moment, the measured value is considered to be linear where in actual the radiation is reduced by the distance squared. An improvement is mentioned in next chapter, Future work.

6 Future Work

Two main areas for the continuous work is suggested.

First of all the most important work needs to be addressed is the further development on TDMA/E-ASAP itself. An improvement with the capability of passing data throughout the network using multi hop technique is recommended. This improves the network and can have the advantage of measuring multiple sources on a larger area of network.

The second and last suggestion for future work is concerned for localisation. A additional functionality is suggested at the GUI application where the measured data can be calculated with different algorithms instantly. A simple solution can be to have a user input that is considered in terms of an algorithm that calculates the position considering this input. This can be a very robust solution where numerous algorithms can be tested instantly without any change in the code of the application.

Appendix A - APIs

A-1 TDMA

TDMA interface for passing through the data using the TDMA and E-ASAP protocols for dynamic wireless network.

Commands

```
void sendData (byte *data)
```

Sets the given data stream for data transmission through the network.

Parameters

data Byte stream for data to be sent. First byte MUST be the length of the following data, i.e If data contains three bytes, the stream must be:
03 XX XX XX.

Events

```
result_t dataReceived(word time, NodeSlotFrame *nsf, byte *data)
```

Signals that data have been received by a node.

Parameters

time Time value when sent by the remote node.

nsf Structure containing the sending node's address, slot and frame. Defined in TDMA.h

data Byte stream of the sent data. First byte contains the length of the data that is followed instantly, i.e If data contains three bytes the stream will be:
03 XX XX XX.

References

[1] Akimitsu Kanzaki, Takahiro Hara and Shojiro Nisho: "An adaptive TDMA slot assignment protocol in ad hoc sensor networks" in Proceedings of the 2005 ACM symposium on Applied computing, Pages: 1160 - 1165 (2005).

[2] Rabbat M., Nowak R. and Bucklew J: "Robust decentralized source localization via averaging" in IEEE International Conference (ICASSP apos;05), Volume 5, Issue , 18-23, Page(s): v/1057 - v/1060, (March 2005).

[3] Qun Li and Rud D: "Global clock synchronization in sensor networks" in Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, Volume 1, Issue , 7-11 Page(s): - 574 (March 2004)

[4] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani: "Clock Synchronization for Wireless Sensor Networks: A Survey" in Computer Science Bibliography, University of Illinois at Chicago, Volume 3, Number 3, Pages 281-323 (May 2005)

[7] TDMA. Available: http://en.wikipedia.org/wiki/Time_division_multiple_access [Accessed March, 2007]

[8] Introduction to TmoteSky. Available: http://rtime.felk.cvut.cz/hw/index.php/Introduction_to_TmoteSky [Accessed March, 2007]

[9] Motiv Community. Available: http://www.moteiv.com/community/Moteiv_Community [Accessed March, 2007]

[10] Wireless sensor network. Available: http://en.wikipedia.org/wiki/Sensor_network [Accessed May 17, 2007]

[11] ZigBee. Available: http://en.wikipedia.org/wiki/ZigBee [Accessed May 17, 2007]

[12] TinyOS. Available: http://en.wikipedia.org/wiki/TinyOS [Accessed May 19, 2007]

[13] The Tinyos Archives. Available: http://mail.millennium.berkeley.edu/pipermail/tinyos/ [Accessed regularly April-May, 2007]