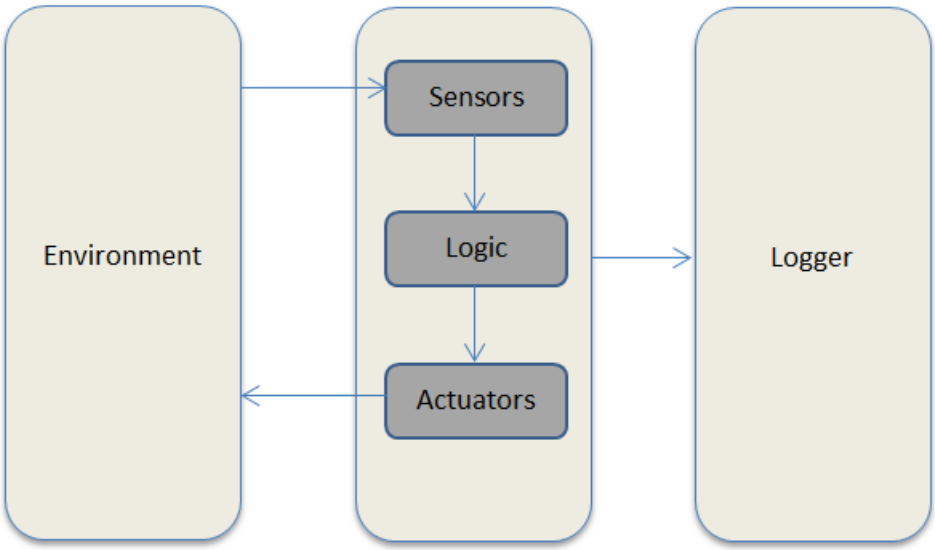
CHALMERS



Precise time-stamping and delays measurement in embedded sensor network

Master's Thesis in Systems, Control and Mechatronics

Georgia-Eirini Diakou

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden

EXE01/2016

Master's thesis EXE01/2016

Precise time-stamping and delays measurement in embedded sensor network

Georgia-Eirini Diakou

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden

Precise time-stamping and delays measurement in embedded sensor network

Georgia-Eirini Diakou

Supervisor: Marina Papatriantafilou, Department of Computer Science and Engineering
Nasser Mohammadiha, Volvo Car Corporation
Examiner: Philippas Tsigas, Department of Computer Science and Engineering

Master's thesis
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2016

Abstract

The field of active safety in the automotive industry has unlimited potentials for the future development of vehicles, especially in autonomy, which is expected to change the way we drive in near future. In recent years, safer cars have become one of the top-priorities among the biggest car manufacturers. The driver support functions that make cars safer rely on accurate onboard sensor readings of the surroundings of the car. To assure the high performance of the active safety systems as well as to avoid problems for their verification during the development stage, it is necessary for the support functions and the sensors to have the same definition of time when an obstacle is detected on the vehicle's surroundings. This requires synchronizing the sensors' data with data from other nodes where sensor fusion and decision making are run for example. Poor synchronization directly inserts unwanted delays between the nodes and impacts the ability of the functions to intervene on time, causing serious safety issues. The present thesis addresses the problem of existence of delays in a real-time embedded system, which consists of different processors without a common timebase, where the tasks that take place are considered as black boxes. The tasks can be, for example, signal processing algorithms. The thesis aims to analyze theoretically the origin and the magnitude of the delays and develop statistical methods for accurate estimations of these delays. The methods are applied to a case-study which consists of an active safety system used in Volvo Cars Corporation. The system includes camera and radar sensors and two processors which process the signals coming from the sensors and make decisions on imminent threats correspondingly. The traffic situations and the vehicle's signals generated at different nodes of the system in response to these situations are logged in an Ethernet-based logging system. Logged data from drivings at public roads and test tracks are used for the analysis. Due to the lack of a common timebase, the developed methods measure the delays based initially on the processors' timebase and then on the logger's timebase. Finally, data coming from a reference sensor system is used to measure the same delays. The overall results of the thesis indicate that the delays depend on the complexity of the traffic environment. In addition, the measurements of delays from the different methods conclude to similar results and the outcome from the reference sensor agree with the outcome from the examined sensor system, which results in a good system's performance.

KEYWORDS: camera, distributed systems, radar, safety-critical systems, sensors, synchronization, timestamps

Acknowledgments

I would like to thank everyone: my supervisor Nasser Mohammadiha and Marina Papatriantafilou and my examiner Philippas Tsigas for the interesting discussions and their support during the work. I would also like to thank the whole group of Active Safety Sensor & Systems Analysis & Verification for their useful input. Additionally, I would like to thank Jury Tarakanov for his help on the subject and his endless ideas.

Georgia-Eirini Diakou, Gothenburg, November 27, 2016

CONTENTS

List of Figures

List of Tables

Notations

Abbreviations

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
CMS	Collision Mitigation Support
ECU	Electronic Control Unit
FCW	Forward Collision Warning
LKA	Lane Keeping Aid
MEMS	Micro-electro-mechanical systems
NTP	Network Time Protocol
RBS	Reference Broadcast Time Synchronization
RRS	Receiver-to-receiver Synchronization
SPI	Serial Peripheral Interface
SRS	Sender-to-receiver Synchronization
TDP	Time Diffusion Protocol
TJA	Traffic Jam Assist
TTC	Time to Collision
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VCC	Volvo Car Corporation
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle

Roman Lower Case Letters

t	timestamp

Greek Letters

Δ	delay

Superscripts

f	the timeline of function processor
l	the timeline of the logging system
s	the timeline of sensor processor

1 INTRODUCTION

Research in sensor networks started at the 1990s, however the field exploded around the year 2000 with the availability of less expensive nodes, sensors and radios (Heidemann and Govindan 2005). The advances in micro-electro-mechanical systems (MEMS) technology contributed to the development of multifunctional sensor nodes equipped with sensing, data processing and communicating components (Rhee *et al.* 2009). During the last decade, the sensor networking has been a very active area with the wide availability of hardware platforms and the growing development of software to make it well-established at research and commercial communities.

1.1 Problem background

Sensor networks are predicated on the ability of sensor nodes to collaboratively detect events, monitor the environment and effect changes to it. A sensor node is depicted in Figure 1.1. The architecture of a sensor node typically consists of a computing device, noted as "logic" in the Figure 1.1, with physical sensors and actuators. The sensors percept the environment and detect changes in it. The sensor data are then fed into the computing device which runs the logic on the data and acts upon the outcome.

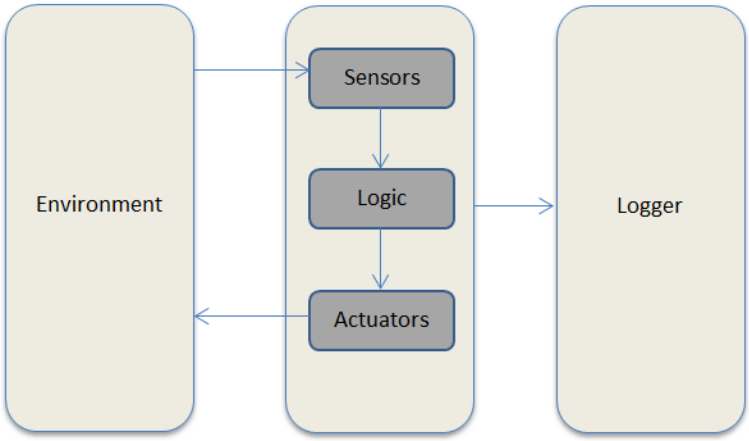


Figure 1.1. A sensor node.

The logic can be consisted of a CPU as well as it is possible to be encompassed by other resources like memory, a human interface, a diagnostic port for diagnosing the system which is being controlled etc. as it can be seen in Figure 1.2. This is an embedded sensor network which can implement a variety of tasks like execute control laws, run finite state machines and signal processing algorithms (Koopman 1996).

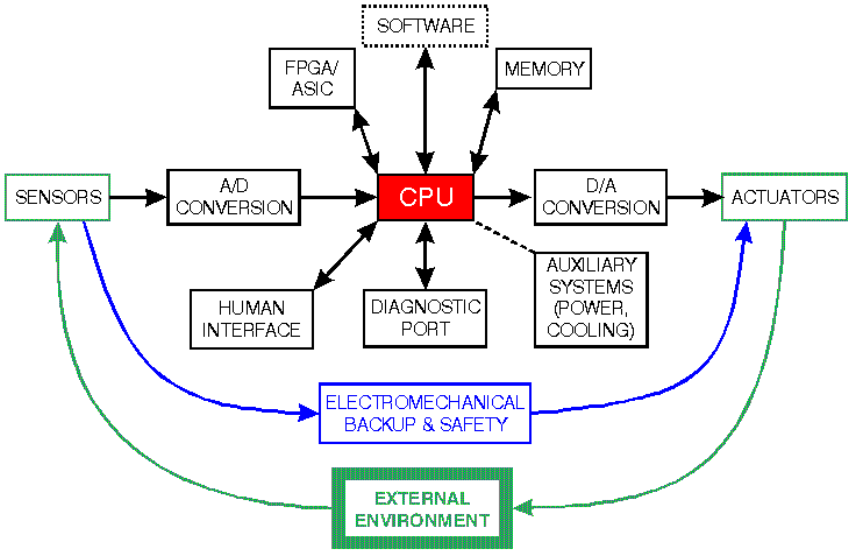


Figure 1.2. Embedded sensor network (Koopman 1996)

Since the embedded sensor networks become more and more complex using large number of sensors, it is important to log all the data from the embedded systems to detect and record faults as well as to develop new technologies and features like new control laws.

In the embedded systems a lot of events take place. An event can be for example the measurement which is taken by the sensor or the transmission of the signal from the A/D conversion to the CPU. The events might be time consuming and this can affect the performance of the system. For instance, if the sensor is a thermometer inside a system which controls the heating system of a building, the time from when the temperature is measured to the moment when the actuator will set the temperature of the heaters in a higher or lower level it does not affect negatively the performance of the system even if this time difference is quite big. However, if the sensor is a camera in a game console which detects the user's movements and has to provide real-time interaction between the user and the game, if the time to process the camera data and provide the actuation is large it will affect the total performance of the system. Except for the last example, there is a big variety of applications based on sensor networks where the time accuracy is a very important aspect and this is the main issue that the present thesis deals with.

The applications of the embedded systems vary from educational applications to industrial, scientific, commercial and environmental deployments, like military applications, environmental monitoring applications, space and automotive applications etc. (Heidemann and Govindan 2005) and timing accuracy is very vital for the progress of all these fields. Specifically, timing accuracy is very crucial for automotive applications in active safety, to increase the vehicle's passengers protection, and for autonomous driving to eliminate the wrong decision-making. In addition,

in the signal processing domain, they deal with the modeling and development of efficient systems for extracting and processing information where their operation is underpinned by precise timing otherwise correct functionality cannot be guaranteed. Telecommunications is another field that timing accuracy is very crucial. They rely on accurate timing to ensure that the switches routing digital signals through networks all run at the same rate. If they did not, those switches running slow would not be able to cope with the traffic and messages would be lost. Finally, applications from global communications to satellite navigation, surveying and transport systems in general depend on precise timing and the same stable and accurate time scale must be in use everywhere for such systems to operate correctly (Stojmenovic 2005). Consequently, the time accuracy is a necessity in sensor networks and there is a big interest towards developing methods and algorithms to provide a common notion of time.

The sensor networks must deal with the need of accurate timing, the synchronization problem among nodes and the lack of a consistent coordinate frame (Zhao and Guibas 2004). Since the sensors in a sensor network can operate independent of each other, their local clocks may not be synchronized and this is an obstructive factor if there is a need of integration and interpretation of information sensed at different nodes. For instance, if a mobile robot is detected at two different times along a path, the times must be comparable and for this reason there should be the possibility to transform the time readings into a common frame of reference in order to define the geometric properties of it, like its speed. Knowing the robot's position at the time when the data was collected is very crucial. Otherwise, and especially if the robot is moving, synchronization errors or failure to map the readings into a common timescale result in projection errors. The faster the robot is moving, the more surprising the magnitude of the error will be, a fact that set serious safety issues (Olson and Arbor 2010). Estimating time differences across nodes with accuracy and reliability is also important in node localization for sensor networks and networks of embedded devices. It uses robust techniques to estimate the range among the nodes providing invaluable context in interpreting sensed data.

1.2 Problem statement

The performance of a sensor network can be restricted by the existence of delays in it which might occur due to the transmission of signals or the heavy signal processing in different parts of it. The present thesis focuses on the measurements of these time delays in a sensor network where the nodes do not have common clocks. To begin with, this requires to identify the sources of delays among the series of nodes taking into consideration the causality of events, the latency occurrence etc. Some important sources of delays are the computationally heavy signal processing on the nodes and the transmission of signals through different interfaces. In addition, it is required to map the nodes' readings into a common timescale and create a congruent understanding of the time among the different clocks in such a way that events which

happen in different nodes should be chronologically comparable.

The ability of the sensor network to timestamp its measurements is used to model the system and create statistical methods for estimating the time differences across the most time consuming tasks on the nodes.

The methods are applied to a case study from the automotive area which is used as a running example through the whole thesis. The system of the case study belongs to the active safety field. The continuing progress in sensing and vehicle technologies has enabled advances in this area which encompasses systems preventing the driver from accidents rather than relieving the accident consequences. An outline of an active safety system is given in Figure 1.3. It is composed of sensors whose role is to perform sensing of the environment and a sensor fusion part which combines and integrates the data derived from the different sensing elements for an improved and robust output. This first part of the active safety system will be called as the *sensors node* for the rest of the thesis. The output is fed to another node which makes decisions on required warnings or interventions to the driver and finally actuation of the braking or the steering system of the vehicle. This node is going to be called as *functions node*.

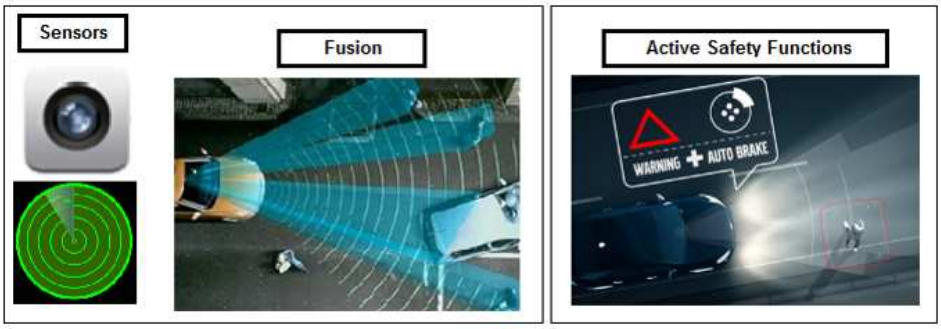


Figure 1.3. Active safety system.

It is of paramount importance to know the exact times when a situation occurs and the related signals generated and sent. The performance of the active safety systems is limited by the fact that every step in signal processing, from detection to actuation, takes considerable, and often uncontrollable, time. The delays come from computationally heavy signal processing as well as transmission of signals between the nodes. Apart from reducing the benefit of the active safety systems, these delays pose problems for their verification during the development stage. It becomes hard to accurately assess their performance, because it becomes impossible to compare the signals with unknown delay with the true traffic situation. In addition, when developing the cars of the future there is a large focus on autonomy. Autonomous cars rely on a network of active safety systems to work and we want them to be able to sense dangers and avoid accidents, without input from the driver. In such a case all the processing times of the different tasks and possible existent delays must be known in order to avoid system failures. Even if in today's cars, especially in

those which move at high speeds, even a delay in the timescale of milliseconds can represent several meters divergence from the actual position of an obstacle. To prove the correct performance of the system and help during the development stage of the system, the traffic situations and the vehicle signals generated in the different nodes are recorded over time through a data logger, Figure 1.4, in order to be analyzed later.

Figure 1.4. Data logger.

1.3 An illustrative example

An illustrative example of the problem is given below. Let us define as *host* the vehicle that we take the measurements and log the data and as *target* every vehicle which is detected by the sensor system if it is inside its field of view. It is also assumed that the longitudinal distance between the host and the target is reported from the two nodes, the sensors node and the functions node, which are shown at Figure 1.3.

The sensors' output about the longitudinal position as well as the output from the active safety functions about the same state estimate are logged and compared. The first node, called as sensor node, reports that the longitudinal position between the vehicles is, for example, $x{=}5m$ at time *13:05:01.100*. The second node, called as functions node, states that the longitudinal position is $x{=}5m$ at time *13:05:01.200*, as it can be seen in Figure 1.5.

There is a discrepancy of *100ms* in the reporting of the time for the same state estimate. If the host's velocity is *20m/s*, a discrepancy of *100ms* results in *2m* uncertainty of the target's position. The higher the host's velocity is, the larger the divergence from the actual position of the target. Let us assume that the actual state estimates calculated by a reference sensor for the time *13:05:01.100* and *13:05:01.200* are shown in Figure 1.6. This time difference of *100ms* might have been occurred due to different factors, one of them can be the delay of the propagation of the signal on the interface between the two nodes. However, this means that the active safety system supposes that the target is *5m* ahead while it is *3m* ahead. A wrong warning

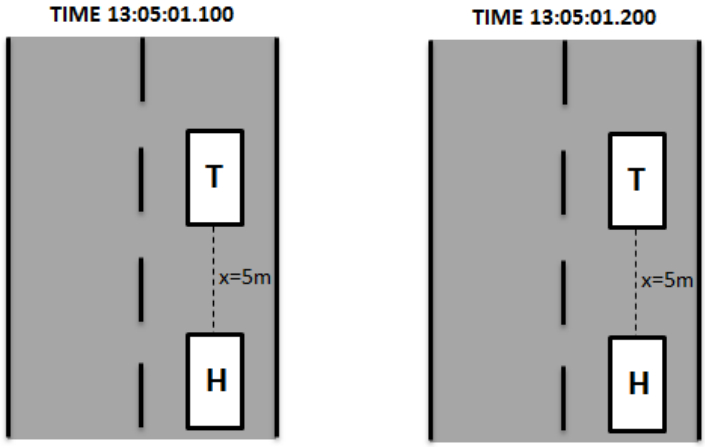


Figure 1.5. The output from a commercial sensor system.

or a late intervention because of this uncertainty will pose problems in the system's performance and will cause safety issues.

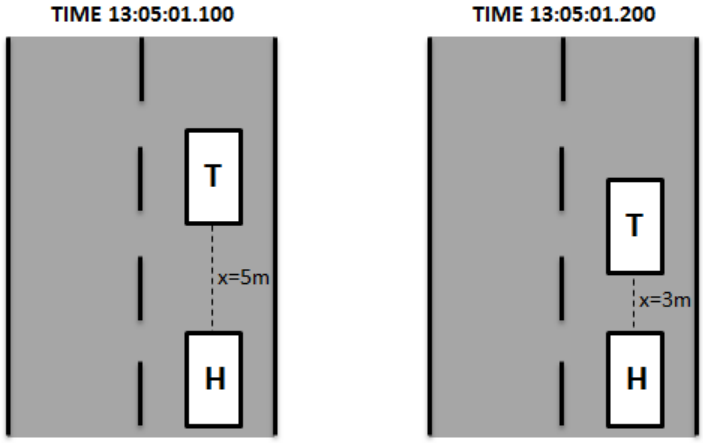


Figure 1.6. The output from a reference sensor system.

A plot from a scenario which is examined in the thesis shows exactly the same problem. For a specific time t, different target's positions are reported from the sensor and the functions nodes as it seems in Figure 1.7. This is caused due to the delays in the system.

1.4 Aim and objectives

The aim of the thesis is to verify empirically the system's performance by estimating theoretically and experimentally the timing accuracy in the embedded sensor network. The system consists of different nodes whose role is to perform sensing of the

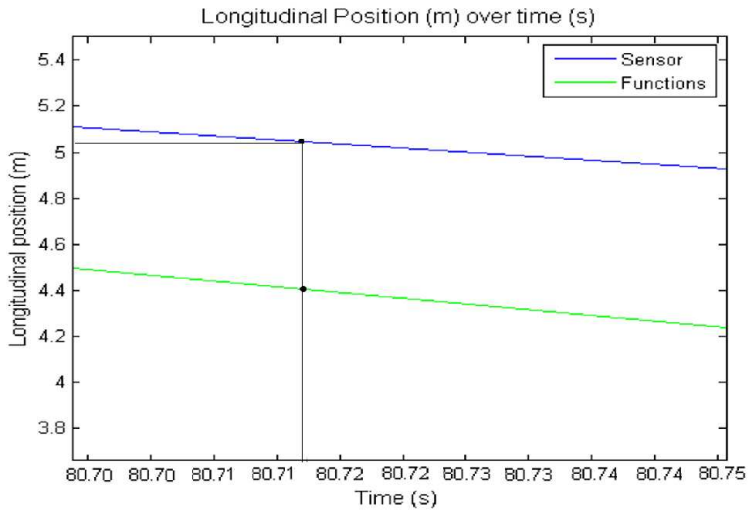


Figure 1.7. Plot showing the existence of delay in a system.

environment as well as to make decisions on required interventions from the driver or the vehicle itself. Each of them has its own clock and is able to timestamp events which take place in the node, for example the sending of a signal from the node to another. However, the sending or receiving of a signal or the signal processing on a node can take a lot of time and as a result delays occur in it. The delays must be measured through the whole signal processing chain of the system with the help of the timestamped events. The focus will be given more to find the sources of delays and create accurate methods to measure them rather than to compensate for them. Since there is not a common reference time basis, a mathematical model will be created to correlate the timestamps and the events from one local timeline to another. Statistical methods will be developed to estimate the delays of the most time consuming tasks of the system. The methods will be evaluated by measuring the delays based on each timeline which exists in the system and compare the final results to each other as well as by the estimation of the same delays using a reference sensor.

1.5 Outline

The report is organized as follows. Chapter 2 presents the current field of research regarding precise time-stamping and delay measurement for time synchronization for sensor networks. Also the theory behind the different methods that are used in the thesis is described. In Chapter 3 an outline of the system which is used in this thesis is given. In Chapter 4 the process of implementing the methods and the different tools used to generate the results is analyzed. In Chapter 5 results are presented. In Chapter 6 an analysis of the results is conducted and finally in Chapter 7 the potential improvements and the continuation of the work are outlined.

2 BACKGROUND

A brief overview of the problems which we fronted during the thesis like the existence of different local clocks in a sensor network and the lack of a common notion of time is described in the Chapter 2. A short summary of important terms and formulas used during the thesis are also included.

2.1 Important definitions

There are a lot of definitions related to clock terminology, synchronization methods and metrics to evaluate synchronization schemes on sensor networks. Most of them are mentioned in the analysis of the thesis' methods while some others are useful to provide a better understanding of them.

Since a sensor network consists of different nodes and it is probable that they do not have access to a common reference clock, their timelines will be different. A *timeline* or an *"arrow of time"* is a progression of time from the past to the future (Kopetz and Ochsenreiter 1987). If the time of a clock in a sensor node A is defined as $C_A(t)$, where $C_A(t) = t$ for a perfect clock, the clock *offset* is the difference between the times that are reported by the clocks of the two nodes and is caused because of the lack of a common time origin of the clocks (Rhee *et al.* 2009). The offset of a clock $C_A(t)$ relative to $C_B(t)$ at a time t is given by $C_A(t) - C_B(t)$. The first derivative of the clock offset with regard to time, $C'_A(t) - C'_B(t)$, is the clock *skew* and is the frequency difference of two clocks. The second derivative of the clock offset with regard to time is the clock *drift* and for two nodes is defined as $C''_A(t) - C''_B(t)$. If we want to map the readings of one node's timeline to another timeline it is important to know those three values, the offset, the skew and the drift.

The *stability* is how well it can maintain a constant frequency (Mills 1991). The *accuracy* is how well its time is compared to national standards and the *precision* is how precisely time can be resolved in a particular timekeeping system. The *reliability* of a timekeeping system is the fraction of the time it can be kept operating and connected in the network.

2.2 Clocks and the synchronization problem

Network sensors consist of a set of nodes which communicate to exchange information. Every node is used to have its own local physical clock. A physical clock can be defined as a device based on some periodic physical oscillation, for example the oscillation of a pendulum, a quartz crystal or an atom, and the local time is the time of a local physical node (Kopetz and Ochsenreiter 1987). Suppose that

a clock of a node i shows time $C_i(t)$ and t is the point of the real time at UTC (Coordinated Universal Time) then a local clock may or may not agree with t. In a perfect clock, $C_i(t) = t$ for all the nodes i and all t. In other words, $dC_i/dt = 1$, in a perfect clock the clock skew is equal to 1. However, the skew depends on different environmental conditions like temperature and humidity and can vary over time. As a result in reality a clock can be faster than the perfect clock if $dC_i/dt > 1$ or slower if $dC_i/dt < 1$. In Figure 2.1 we can see the behaviour of a perfect, a fast and a slow clock. Nonetheless, it can be assumed that a clock is within its specifications if:

$$1 - \rho \leq \frac{dC_i}{dt} \leq 1 + \rho, \tag{2.1}$$

where ρ is the maximum skew rate.

The synchronization problem corresponds to the problem of equalizing the clocks of different nodes. This is not an easy problem to solve as the clocks of the nodes drift away over a time period, because of the temperature changes etc., and simultaneously there is a need of equalizing the offset and the clocks' rates or repeatedly correct the offsets every specific interval to keep the clocks synchronized. For two nodes that they have hardware clocks with stability of ± 5 ppm (parts per million), they can drift by as much as 10μs within one second (Hazra *et al.* 2009). The existence of synchronization though among the nodes in a sensor network is very vital in order to exist a common understanding of the environment.

The synchronization problem can have different forms (Sivrikaya and Yener 2004). The first one corresponds to ordering the events or the messages and just say if the event *a* occurs before the event *b*. This is the simplest form of synchronization and it is more a comparison of local clocks for order rather than time synchronization. The second one includes the algorithms that keep track of the relative offset and drift among the local clocks of nodes and they are able to convert the instant time of a node to some other node's local time. Finally, the third category includes synchronization algorithms that all nodes' clocks are synchronized to a reference clock in a common global timescale (Sivrikaya and Yener 2004). In this case the local clocks are continuously adjustable to the reference clock (Flaviu 1989). If the local clock is $C_L = L$ while the reference clock C_R shows time R and $L \neq R$, the goal is to increase the speed of the local clock if $R > L$ or decrease it if $R < L$ in order to show time $R + \alpha$ instead of $L + \alpha$, where α is an amortization parameter. The reference clock is a stable clock, usually Cesium or Rubidium based stable automatic clocks,

2.3 The need for synchronization in sensor networks

There are two main types of sensor networks: Centralized and distributed systems. In the first category, there is a clear ordering of the events and the times at which the

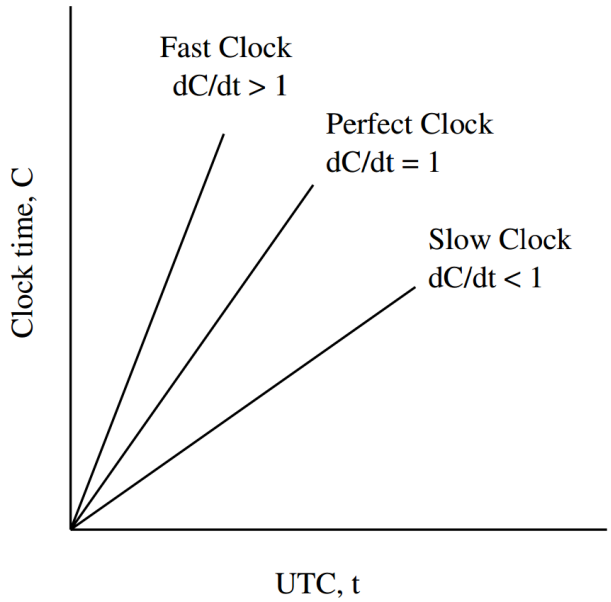


Figure 2.1. Illustration of fast, perfect and slow clock (Sundararaman *et al.* 2005).

events happen are known because there is a common global timeline. One process is timestamped and every other process which tries to get a timestamp, it will get an equal or a higher value of time. However, in distributed systems, each processor has its own internal clock and there is not a common notion of a time as there is not a common global clock. In this case, there is an increased need for synchronization.

Since the sensor networks are mostly distributed systems, there are several reasons for addressing the need of synchronization in them. One application that exact timestamping and synchronization are indispensable is the multi-sensor fusion (Huck *et al.* 2011), (Westenberger *et al.* 2011), (Brahmi *et al.* 2013). Sensor nodes need to collaborate and coordinate their operations in order to achieve sensing tasks and end up to a meaningful result. Especially, the active safety systems are based on different sensors e.g. cameras, radars, laser-scanners etc. that are fused to associate their data and provide an accurate perception of the environment. For example, in tracking applications, sensor nodes report the location and time at which they sense the vehicle to a sink node where information is combined to estimate the location and velocity of the vehicle. Incorrect timestamps lead to wrong estimations of the position of the detected vehicles and traffic situations cannot be evaluated and rated correctly. Especially, in the case of very time critical applications like in pre-crash situations or automatic emergency braking or steering, where the decisions must be made in milliseconds, different latencies affect negatively the estimation of the real time of measurement. Obviously these latencies cannot be neglected but instead should be detected and compensated. Other applications are the vehicle-to-vehicle (V2V) and the vehicle-to-infrastructure (V2I) communication where different vehicles communicate the results from their local sensor fusion to other vehicles or

to infrastructure. In these specific cases there are several problems that should be taken into consideration and this is that not only the local sensors have to be synchronized but also the clocks of the different vehicles and of the infrastructure.

2.4 Synchronization protocols

For the synchronization of nodes in different sensors networks there are a lot of algorithms that have attracted a lot of attention in recent years. Some of them rely on clock information from the GPS (Global Positioning System) and high power receivers which are appropriate for a lot of networks, however they can present disadvantages for others because of their cost, their size and their non-availability in some situations like underwater, under foliage etc. If those algorithms cannot be used in a network for the time synchronization, software-based protocols are developed which achieve the same result. The choice of the synchronization protocol depends on the application and the characteristics of a network so the protocols are different from each other in some aspects but very similar in some others. A classification of the synchronization protocols that can give an understanding of the variety and the potentials that they have can be done based on either synchronization issues or based on application features which are discussed in the following sections (Rhee *et al.* 2009), (Sundararaman *et al.* 2005).

2.4.1 Classification of the synchronization protocols based on synchronization issues

Initially, a classification of the software-based protocols is done based on synchronization related issues. This classification includes the master-slave and the peer-to-peer synchronization, the internal and external synchronization, the probabilistic and deterministic synchronization, the sender-to-receiver and receiver-to-receiver synchronization, the clock correction and untethered clocks and the pairwise synchronization versus the network-wide synchronization (Sundararaman *et al.* 2005).

The *master-slave* protocol assigns one of the nodes as the master and the rest of them as slaves (Sundararaman *et al.* 2005). The local clock reading of the master node is assumed to be the reference time and the slaves are trying to synchronize with the master. On the other hand, the *peer-to-peer* structure allows any node to communicate directly with all the other nodes of the network. This removes the risk of the master node failure making the structure more flexible but at the same time more uncontrollable (Sundararaman *et al.* 2005). The structure which will be used in the thesis is the peer-to-peer structure since there is not a master node.

In *internal* synchronization there is no global time base available and for this reason the protocol attempts to minimize the maximum difference between the readings of local clocks of the sensors (Sundararaman *et al.* 2005). The internal synchronization

can be performed in master-slave or peer-to-peer structures. In *external* synchronization, a source of time like UTC, is available and is used as the reference time. The local clocks of the sensors are adjusted to this time in order to be synchronized. External synchronization uses phase locked oscillators in order to maintain the local clocks within a specified deviation from the externally maintained time reference. In the contrary to the internal synchronization, the external is not possible to perform in peer-to-peer structure as it requires a master node to communicate with a time service like GPS to synchronize the slaves and itself to the reference time (Sundararaman *et al.* 2005). In the examined sensor network of the present thesis there is neither a master node nor a GPS system so only internal synchronization methods are taken into account.

In *Sender-to-receiver* synchronization, SRS, the sender sends periodically a timestamp with its current local time information to the receiver and the message delay is calculated as the total round-trip time from when the receiver requires the timestamp to the time that it receives it (Sundararaman *et al.* 2005). Apparently, in this method there is a variance in message delay between the sender and the receiver due to network delays and workload in the nodes. The *Receiver-to-receiver* synchronization, RRS, uses the time that each receiver receives the same messages and it does not require the sender's participation at all. In this approach it is assumed that if any two receivers receive the same message in single-hop transmission, they receive it at approximately same time. The receivers then calculate the offset among the times that they have received the same message. This kind of synchronization reduces the variance of the message delay (Sundararaman *et al.* 2005). This thesis deals with an SRS structure rather than an RRS.

In the *clock correction* approach, the network's local clocks of nodes are corrected either instantaneously or continually to run in parallel with a global timescale or an atomic clock which provide a reference time to keep the entire network synchronized (Sundararaman *et al.* 2005). In the approach of *untethered clocks* a common notion of time among the nodes of a network is achieved without synchronization but instead a table of parameters is created which are used to relate the local clock of each node to the local clock of every other node in the same network. The time translation helps to compare all the local timestamps (Sundararaman *et al.* 2005). In the present thesis, the sensor networks which are examined follow the untethered clocks principle since there is not a global timescale.

In addition, the *pairwise* synchronization is used to synchronize two nodes while the *network-wide* synchronization is designed to synchronize a large number of nodes in the network (Sundararaman *et al.* 2005). The examined sensor networks in the thesis deal with the pairwise synchronization.

2.4.2 Classification of the synchronization protocols based on applications features

This section describes the classification of software-based protocols based on applications features. The classification includes the single-hop and multi-hop networks, the stationary networks and mobile networks and finally the MAC-layer-based approach versus standard approach (Sundararaman *et al.* 2005).

In a *single-hop* network a sensor node can directly communicate and exchange messages with any other node which belongs to the network. On the other hand, in the *multi-hop* communication the sensors in a domain communicate with sensors in another domain via an intermediate sensor relating to both domains (Sundararaman *et al.* 2005). In this thesis' sensor network, there is no intermediate node but the nodes can directly communicate and exchange messages.

In *Stationary* networks the sensors do not have the ability to move. However, in *mobile* networks the sensors are not stationary but they move and they have the ability to connect with other sensors when they enter geographically the area of these sensors. Thin means that the topology is changing often and it might insert problems as the nodes should be synchronized again with each other (Sundararaman *et al.* 2005).

3 SYSTEM DESCRIPTION AND MODELING

In Chapter 3 the problem of existing delays in a sensor network is analyzed, the challenges are presented and the system which is exposed to delays is modelled. In addition, an outline of an embedded sensor and processor network is given which is used as a running example throughout the whole thesis.

3.1 Modeling of the system

3.1.1 Source of delays

To measure the time delays which exist in a sensor network where the nodes do not have common clocks requires first to identify the sources of delays among the series of nodes. However, this is a procedure which encompasses a lot of challenges. Often, time synchronization methods among nodes are applied to sensor networks and they are accomplished by exchanging messages. These messages allow to compute time differences among the clocks and help a node to estimate the time in other nodes' clock. Having calculated the time differences, it is possible to adjust or correct the clocks to operate in tandem. A challenge though is that the messages are exposed to different delays while they are starting from a source node and reaching to its destination through different paths. Especially non-deterministic and unbounded delays make synchronization difficult. However, in certain applications like real-time automation and active safety functions, it is very vital to keep a certain maximum level of latency because otherwise the application will not work in a satisfactory manner or in the worst case it can fail outright. The most significant delays are the nodal processing delay, the queuing delay, the transmission delay and propagation delay and all together are accumulated to give a total nodal delay. The performance of Active Safety systems is greatly affected by network delays so it is vital to know the nature and the importance of these delays (Kurose and Ross 2013).

Let us assume that the sensor network that we examine consists of two nodes as the Figure 3.1 shows.

The aim is to be able to calculate the delay x from when, for example, the signal sent from node A to when it was received from node B. This delay might contain delays which take place inside the nodes.

In a general system of two nodes a packet is transmitted from the source node A through an outbound link to the destination node B. This link is probably preceded by a queue (known also as buffer).

Figure 3.1. The nodes of a sensor network.

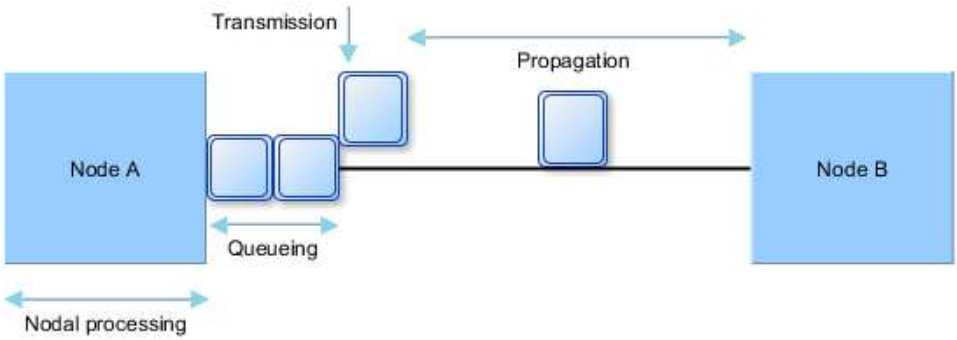


Figure 3.2. The delays through two nodes.

- Processing delay

The processing delay, d_{proc}, includes the time which is required to process the packet's header and determine the packet's destination. It may also include factors like the time needed to check for bit-level errors in the packet that occurred in transmitting the packet's bits from a previous node to node A. Processing delays are typically on the order of microseconds or less.

- Queueing delay

After the nodal processing is completed, the source node A directs the packet to a queue that precedes the link to node B. While the packet waits to be transmitted onto the link it is exposed to the queueing delay d_{queue}, which depends on the number of earlier-arriving packets which are queued and waiting for transmission across the link. The delay of a packet can vary significantly. If no other packet is currently transmitted, the queue is empty and the queueing delay is zero, but if there is a lot of traffic and other packets are waiting to be transmitted the queuing delay will be long. Queuing delays can be on the order of microseconds to milliseconds in practice.

The queueing delay is the most complicated type of delay because it can vary from packet to packet. For example, if 10 packets arrive at a queue at the same time

and the queue is empty, the first packet will be exposed to no d_{queue} whereas the last one will suffer from a relatively large d_{queue} waiting for the previous packets to be transmitted. Calculating with certainty the queuing latency requires a detailed knowledge about all sources of traffic on the network. Due to the fact that the queuing delay is very variable it is preferable to use statistical methods to characterize it, like the average, the variance or the probability that the queuing delay exceeds a specific value.

- Transmission delay

The transmission delay d_{trans} is the time needed to push all the packet's bits into the link. If it is assumed that the packets are transmitted in a first-come-first-served manner, the packet can be transmitted only after all the packets that have arrived before it have been transmitted. The transmission delay does not depend on the distance between the nodes but on the packet's length in bits and the transmission rate of the link. If we define the length of the packet by L bits and the transmission rate by R bits/sec, then the transmission delay is defined as

$$d_{trans} = \frac{L}{R} \tag{3.1}$$

Transmission delays are typically on the order of microseconds to milliseconds in practice.

- Propagation delay

The propagation delay d_{prop}, which is the last delay that the packet is exposed, is the time that it needs to travel over the link and reach at the destination node. The bits propagate at the propagation speed of the link and the propagation speed depends on the physical medium of the link. The physical link can be for example fiber optics, twisted-pair copper wires etc. On the contrary to the transmission delay, the propagation delay depends on the distance between two nodes. If we define the distance between node A and node B as d and the propagation speed of the link as s, the propagation delay is the distance between them divided by the propagation speed as it is defined by the Equation 3.2

$$d_{prop} = \frac{d}{s} \tag{3.2}$$

Typically the propagation delays are on the order of milliseconds.

- Total nodal delay

The contribution of these delay components can vary significantly. The sum of the processing, queueing, transmission and propagation delay gives the total nodal delay by:

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop} \tag{3.3}$$

In an Ethernet-based network for example, some sources of delay are calculated. For the maximum size Ethernet frame 1500 *bytes* at rate 100 $Mbps$ the transmission delay is 120 μs. For comparison, the minimum size frame 64 *bytes* at rate of 1 $Gbps$ has a latency of 0,5 μs.

In addition, in the Ethernet network the bits which are transmitted along a fiber link travel at $2 \cdot 10^8$ m/s. So, the one way latency for 10 km link is 50 μs using the Equation 3.2.

Finally, the queuing delay is variable from packet to packet depending on the load of network. If we want to calculate the average queueing delay of the Ethernet network then we should use the Equation 3.4

$$d_{queue} = (network\ load) * d_{trans,(max)} \tag{3.4}$$

The average queuing latency is proportionally to the network load. The network load is given as the fractional load relative to full network capacity and the $d_{trans,(max)}$ is the transmission delay of a full-size frame. If we assume that the network has 25% load and the frame is 1500 *bytes*, the average queueing delay of the Ethernet network is 30 μs.

In some circumstances, packet loss occurs when one or more packets of data travelling across the network fail to reach their destination. This happens due to the fact that in reality the queue preceding a link has finite capacity so a packet can arrive to find a full queue. With no place to store the packet, a node will drop it and the packet will be lost. A packet loss looks like a packet having been transmitted into the network core but never emerging from the network at the destination. Therefore, performance at a node is often measured not only in terms of delay, but also in terms of the probability of packet loss.

3.1.2 Events

Those were the most common delays which can be found in systems. Different events though take place in different systems. To be able to define the delays in every given system irrespectively the complexity of the system it is necessary to define events.

It is assumed that the examined system is composed of different processes P_i, i=1,2,...x. Each process consists of a sequence of events. The events of a process form a sequence where event a occurs before event b in the sequence if a happens before b. The type of events depends on the application we have. However, in distributed systems it is sometimes hard to say that one of two events occurred first (Lamport 1987). We assume though a partial ordering of events inside the system where the partial ordering is defined by an arrow $\rightarrow$ and it is applied if:

i. If a and b are events in the same process and a comes before b then $a \rightarrow b$.

ii. In addition, if a is the sending of a message by one process and b is the receipt of the same message by another process then $a \rightarrow b$.

iii. Finally, if $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$.

For example, according to the system of Figure 3.2, a is the sending of a signal from Node A and b is the receiving of the signal from Node B and according to the definition (ii) it will be $a \rightarrow b$.

3.1.3 Timestamps

We now introduce clocks into the system. According to (Lamport 1987), 'a logical clock is just a way of assigning a number to an event, where the number is thought of as the time at which the event occurred'. The challenge though, as it was mentioned before, is that each node has its own clock and they do not have access to a central clock which requires a service to ensure that the processors share a common notion of time, where time can mean either an approximation to real time or an integer-valued counter. In case that there was a common clock, the nodes could assign timestamps to events in this common global time and the delays would be easily measurable by a simple subtraction of the timestamps. However, especially in distributed systems, it may be ineffective to use external clocks, as the cables interconnecting these clocks introduce distortions, which could be higher than the inherent stability of the external clock. In addition, the connection of each node to a stable clock is a very expensive solution which is not preferred very often.

In this case though the non existence of a common time base creates the need of applying internal synchronization methods among the nodes. These methods are software based solutions which ensure that the logical clocks used by the nodes are consistent within limits and they provide the possibility of common interpretation of information. The software algorithms assure that the comparison of the timestamps of specific events between nodes are able to provide numerical answers about the delays within the system. In addition, we should remember that physical clocks may drift due to temperature changes etc. as we saw in Chapter 2. The clocks in our system might also drift.

Let $C_i(t)$ denote the reading of a clock C_i, i=1,2,...x depending on how many different clocks exist in a system, at physical time t.

Now, we define each clock $C_i(t)$ for each process P_i to be a function which assigns the current time t which it reads from its clock C_i to an event a in that process. This results in the fact that each event at a process P_i will be attached with a timestamp t_i equal to the time that it reads from its clock C_i when the event happens. Each clock defines also a unique timeline. Given that we have x clocks in the system it means that we will have x timelines respectively.

The following example sums up what it was mentioned till now about the delays, events and timestamps. It was defined at the Section 3.1.2 that the event a is the sending of a signal from Node A and the event b is the receipt of it from Node B. The Node A has its own clock which is the C_A and the Node B has another clock defined as C_B. Two timelines exist because of the two different clocks, the first is the Node's A timeline based on its hardware clock C_A as well as the second is the Node's B timeline based on C_B. Because of the two different clocks and as there is a lack of a common time origin of them, there are offsets among the clocks. The offset between the C_A and C_B clocks is defined as $\Delta(C_A, C_B) = C_B - C_A$. As a result, in order to introduce the timestamps in the modelling of the system, the $t_A < a >$ is the timestamp which is attached by the clock of Node A C_A to the sending of the signal from Node A. Respectively, $t_B < a >$ is the current time which is assigned from the clock C_B to the event a. When the event a happens it is possible to be timestamped by the clock C_A since it is a part of node A however it is also possible to be timestamped by the C_B if and only if C_A is synchronized with C_B. As a result, it is possible the two clocks to assign a timestamp for the same event only if the nodes are synchronized.

Finally, to measure the delay x which is shown in the Figure 3.1, which is the delay from when the signal is sent from node A to when it is received from node B, we need to solve the next equation:

$$x = t_B < b > -t_B < a > \tag{3.5}$$

However, if there was a central clock and all the nodes had access to that, then it would be enough to solve:

$$x = T < b > -T < a > \tag{3.6}$$

if it is supposed that T is the global time, e.g. the UTC time.

Modelling the delays of the whole system with equations like the 3.5 and solve them will give us the solution to our problem. However, if we do not have the possibility to interfere and verify the synchronization method among the nodes we have to accept that it is accurate enough even if it might be a possible source of delays. For this reason and in order to extinguish the uncertainties we will develop statistical

methods to measure the delays. The methods will be applied to a large sample of data and they will measure the delays based on different timelines. This means that the delay x will be measured based on the timestamps that the clock C_A reports but also based on the timestamps that the clock C_B reports. The measurements will be compared to verify if the results are similar. What is more, a reference system can be used to provide results about the same delay.

3.2 Case study

A use-case system is modelled according to the previous analysis and is used as a running example throughout the thesis. It includes an embedded sensor network which is used by Volvo Cars Corporation (VCC) for the development and verification of advanced driver assistance systems (ADAS).

The system consists of sensors with radar and vision sensing for aiding the car's perception of its surrounding. It can access an imminent threat and enable a suite of active safety features like forward collision warning (FCW), lane keeping aid (LKA), collision mitigation support (CMS) etc., if there is a need. The system's sensors are located in one processor where they detect the obstacles, the data is fused and then is sent over an interface to a second processor which is responsible to make decisions and apply the active safety functions, if it is necessary. The signals through the whole chain detection-fusion-decision are converted to UDP packets and sent through a dedicated Ethernet network to a logging device. The two processors and the logger have each of them local clocks and while the signals are timestamped during the different procedures in the signal processing chain, any error affecting the timestamp could have a direct impact on the active safety functions and the accuracy of the system. It is possible to have delays in both data processing and the logging system which result in the failure of the accurate correlation of signals to each other and to the real traffic situations.

The system is depicted in Figure 3.3 and consists of a sensor processor and a function processor that are connected through a Serial Peripheral Interface (SPI) data bus. SPI is often used for transferring data between processors for several reasons, e.g. it covers full duplex communication, it is not limited to 8-bit words in the case of bit-transferring, it has a simple hardware interfacing and it is cheap. The data from the two processors is converted to UDP packets and sent through a dedicated Ethernet-based UDP network to a logging system.

The sensor processor includes integrated sensors for radar sensing, vision sensing and data fusion. It uses data fusion algorithms to combine inputs from the sensors to deliver high-accuracy estimates of the parameters describing the environment. The function processor consists of further processing of the sensor data from the sensor processor but also other sensors. This module is named as pre-processing for functions and its output is fed as input to the driver's support functions, like adaptive cruise control (ACC), traffic jam assist (TJA) etc. where decision making is

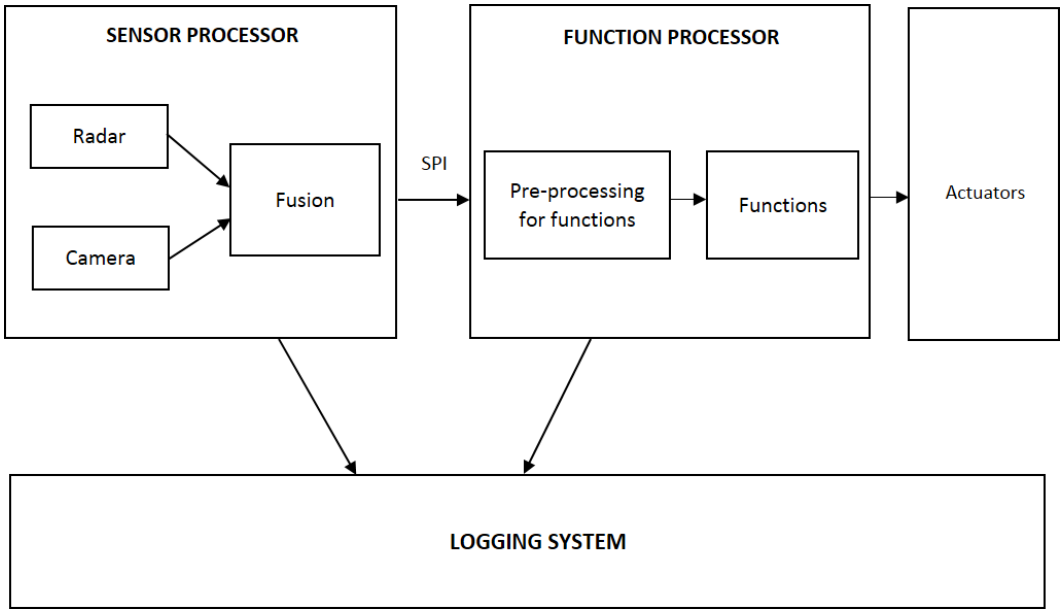


Figure 3.3. Block diagram of the use-case system.

taking place. The decisions are implemented in the actuators. The logging system records the data at different points. This data is used for the verification of the correct performance of the sensors and the functions.

In this distributed system, the sensor data acquisition is carried out in one processor and the algorithms for the functions run in the second processor. Because of this fact there is a delay for the sensor data to propagate to the functions. The magnitude of the delay depends on the processing times of the different tasks that they take place in the meantime as well as on the transmission of signals through the SPI. It is very vital for the functions to know the exact state estimates (position, velocity, acceleration) of the target the moment that they should make a decision and brake or steer. However, the delay inserts uncertainty of the obstacles' state estimates when the data reaches the functions. For example, if the sensor data detects that the target is at a specific position at a specific time but this information become known to the functions 100ms later, during this time both the host and the target car have been moved and if the functions decide how to act based on the delayed data, their performance will not be satisfactory. The solution is to measure this delay and compensate for it. While a compensation is made in the current system, this thesis aims to measure the accuracy of the delays by defining different methods to calculate the means and distributions of them. This requires in depth analysis of the system and good handling of the inherent variable which is the time.

3.2.1 Definition of delays

Every step in the signal processing chain from the obstacle detection by the sensors to decision making in the functions takes considerable time because of the computationally heavy signal processing methods on the nodes and the transmission of signals through different interfaces, e.g. the SPI. To capture possible delays in the system, we aim to estimate mathematically the processing times of the time-consuming tasks. Those tasks are the sensor detection and fusion, the transferring of the data over the SPI and the decision making at the functions. A scheme of the times are given at the Figure 3.4. The times we aim to estimate are the Δt_{sens}, Δt_{trans}, $\Delta t_{sens+trans}$ and Δt_{funct} as it seems in Figure 3.4.

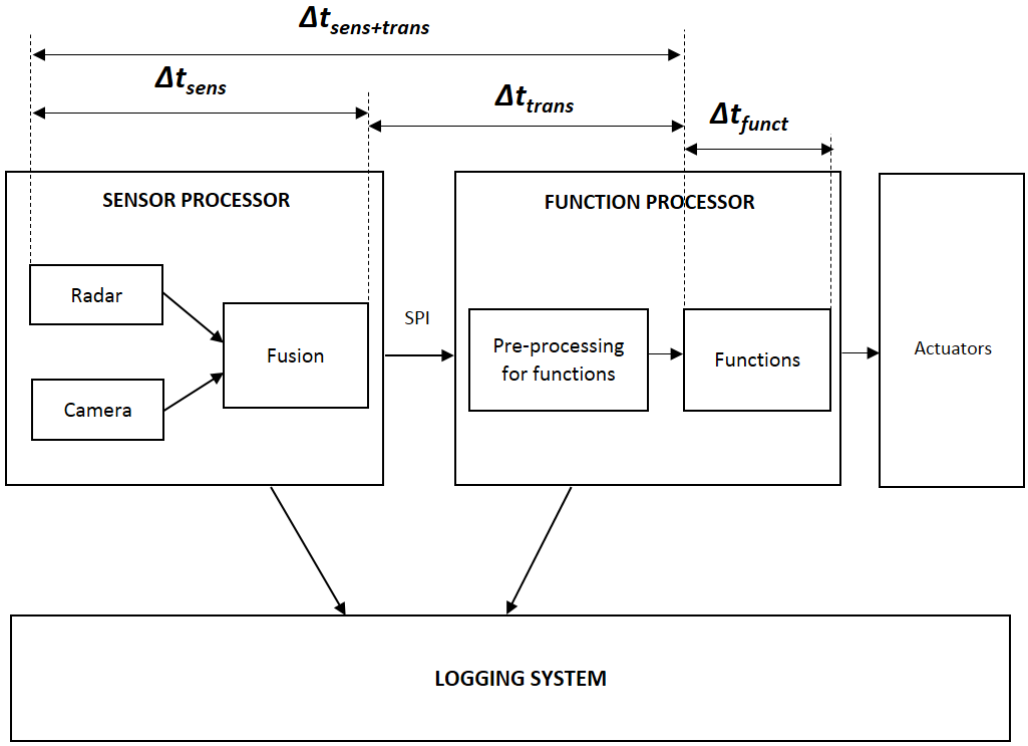


Figure 3.4. Processing times of time consuming tasks.

The times we aim to estimate are summarized below:

- Δt_{sens} is the processing time of the sensors and the fusion in total.
- Δt_{trans} is the time it takes to transfer the data trough the SPI as well as the time in the pre-processing for functions module.
- $\Delta t_{sens+trans}$ is the sum of the first two measurements.
- Δt_{funct} is the processing time in the functions.

To proceed with the estimation of the delays sensors and functions data is required from the case-study system. The data was collected by Volvo Car Corporation during test runs. The dataset contains data from the radar and camera sensors as well as ground-truth data from a reference system which is installed on the car.

3.2.2 Clocks and timelines

The present distributed system consists of two processors that communicate by messages transmission through the SPI interface and each processor has its own hardware clock without access to a central clock.

Since in this system the sensor data acquisition and the algorithms for the functions run in different processors a synchronization method to ensure that the processors share a common notion of time is very vital to reduce timing error versus naively timestamping sensor data when it arrives at the function processor.

Except for the two hardware clocks of the processors, the logging system, run in a separate processor, has also a different clock and as a result there are three different timelines in the system:

i. The sensor processor's timeline based on its hardware clock C_s.

ii. The function processor's timeline based on its hardware clock C_f.

iii. The logger's timeline based on its clock C_l.

Because of the three different clocks and as there is a lack of a common time origin of them, there are offsets among the clocks. The offset between the C_s and C_f clocks is defined as $\Delta(C_s, C_f) = C_f - C_s$ and accordingly $\Delta(C_s, C_l) = C_l - C_s$, $\Delta(C_f, C_l) = C_l - C_f$ are the other two offsets.

The different timelines of the system is shown in Figure 3.5 and the relationship among them is given by the Equation 3.7.

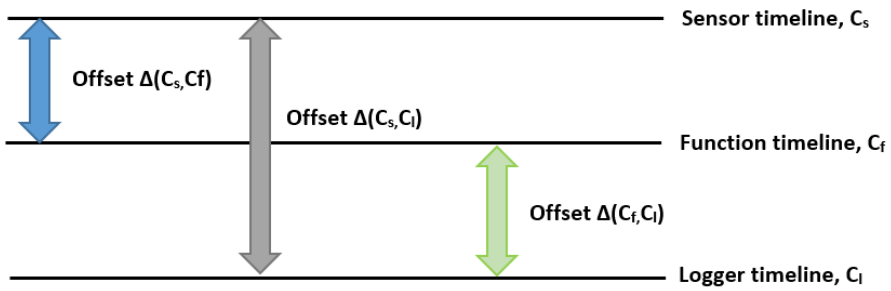


Figure 3.5. Timelines in the system.

$$\Delta(C_s, C_l) = \Delta(C_s, C_f) + \Delta(C_f, C_l) \tag{3.7}$$

The offset between the function and logger timeline is the sum of the offset between the sensor and logger timeline and the offset between the sensor and function timeline. We have the following cases:

- $\Delta(C_f, C_l) > 0$ if $C_l > C_f$
- $\Delta(C_s, C_l) > 0$ if $C_l > C_s$
- $\Delta(C_s, C_f) > 0$ if $C_f > C_s$

3.2.3 Events and timestamps

The use-case system is composed of three processes P_s, P_f and P_l. The P_s is the sensor processor, the P_f is the function processor and the P_l is the logging system.

Different events take place inside the processes and the most important are listed below:

e1. Radar measurement
e2. Radar measurement sent to logger
e3. Radar measurement received at logger
e4. Camera measurement
e5. Camera measurement sent to logger
e6. Camera measurement received at logger
e7. Fusion output
e8. Fusion output sent to functions
e9. Fusion output sent to logger
e10. Fusion output received at functions (as input to the pre-processing)
e11. Fusion output received at logger
e12. Functions input
e13. Functions output
e14. Functions output sent to logger
e15. Functions output received at logger

Now, partial ordering relations among the events can be defined according to the Section 3.1.2 on when one of two events occurred first. The abbreviation $meas = measurement$ is used for sake of simplicity.

- For example, $< radar\ meas >$ and $< radar\ meas\ sent\ to\ logger >$ are events in the same process and $< radar\ meas >$ happens before $< radar\ meas\ sent\ to\ logger >$ so according to (i):
  $< radar\ meas > \rightarrow < radar\ meas\ sent\ to\ logger >$.

- According to the second definition, $< radar\ meas\ sent\ to\ logger >$ is a sending of a message by one process and $< radar\ meas\ received\ at\ logger >$ is the receipt of the same message by another process so:
  $< radar\ meas\ sent\ to\ logger > \rightarrow < radar\ meas\ received\ at\ logger >$.

- Finally, since it is $< radar\ meas > \rightarrow < radar\ meas\ sent\ to\ logger >$ and $< radar\ meas\ sent\ to\ logger > \rightarrow < radar\ meas\ received\ at\ logger >$ according to the definition (iii) occurs that:
  $< radar\ meas > \rightarrow < radar\ meas\ received\ at\ logger >$.

Similarly, the next logical relations occur:

- $< camera\ meas > \rightarrow < camera\ meas\ sent\ to\ logger >$
- $< camera\ meas\ sent\ to\ logger > \rightarrow < camera\ meas\ received\ at\ logger >$
- $< camera\ meas > \rightarrow < camera\ meas\ received\ at\ logger >$
- $< fusion\ output\ sent\ to\ functions > \rightarrow < fusion\ output\ received\ at\ functions >$
- $< fusion\ output\ sent\ to\ logger > \rightarrow < fusion\ output\ received\ at\ logger >$
- $< functions\ input > \rightarrow < functions\ output >$
- $< functions\ output > \rightarrow < functions\ output\ sent\ to\ logger >$
- $< functions\ output\ sent\ to\ logger > \rightarrow < functions\ output\ received\ at\ logger >$
- $< functions\ input > \rightarrow < functions\ output\ sent\ to\ logger >$
- $< functions\ input > \rightarrow < functions\ output\ received\ at\ logger >$
- $< functions\ output > \rightarrow < functions\ output\ received\ at\ logger >$

However we cannot say anything about the relation between the camera events, e.g. $< camera\ meas >$ and the radar events $< radar\ meas >$. There is no order between them, the one can proceed the other and vice-versa depending on their sampling rate.

As it was mentioned, three clocks exist in the system the C_s, C_f and C_l which assign timestamps to the events. For example, the timestamp for the event $< radar\ meas >$ is defined as $t_s < radar\ meas >$. Respectively, for the event $< functions\ input >$ at process P_f, the timestamp which is assigned is $t_f < functions\ input >$. According to what it was just mentioned, all the events which were defined before they should be assigned with timestamps from the processors that they belong to. So, events e1, e2, e4, e5, e7, e8 and e9 are assigned timestamps from C_s. Events e3, e6, e11 and e11 are assigned timestamps from C_l and events e10, e12, e13 and e14 are assigned timestamps from C_f.

Some of the timestamps are not logged, so there is not access to them like the e8, e10, e13 and e14. The overall timestamps are summarized in Table 3.1.

However, the question in this point is how it is possible to correlate and find relations among the timestamps from the different processors and as a result calculate the wanted delays. The answer is given by a synchronization method between the sensor and function processor which is based on a synchronization pulse. Using this method, an algorithm is implemented to transfer $t_s < radar\ meas >$ to the

Table 3.1. Timestamps assigned at the events.

Events	Timestamps of events
Radar measurement	t_s <radar meas>
Radar measurement sent to logger	t_s <radar meas sent to logger>
Radar measurement received at logger	t_l <radar meas received at logger>
Camera measurement	t_s <camera meas>
Camera measurement sent to logger	t_s <camera meas sent to logger>
Camera measurement received at logger	t_l <camera meas received at logger>
Fusion output	t_s <fusion output>
Fusion output sent to logger	t_s <fusion output sent to logger>
Fusion output received at logger	t_l <fusion output received at logger>
Functions input	t_f <functions input>
Functions output received at logger	t_l <functions output received at logger>

$t_f < radar\ meas >$. This synchronization method assures also that the times indicated from different clocks do not drift arbitrarily far apart and the clocks of the sensor and function processors are consistent. In other words, the timestamps produced by the C_s clock when the radar measurement is taken were already known but after the synchronization the timestamps that would be given by the C_f clock when the radar measurement is taken become also known. So, it becomes known the $C_f(t)$ value that it would be given to the event $< radar\ meas >$ if it was observable by the function processor. This relation is the key for the solution of the problem since it creates the foundation to compare timestamps between the two processors. This relation is used to create a mathematical model with equations which correlate additionally other timestamps from all the processors together and are used later to calculate the delays.

3.2.4 Mathematical modeling of the case-study

Having defined the events and the timestamps in the different timelines we aim now to create the equations or else the mathematical model to calculate the delays. The equations will be a comparison between the timestamps for example to calculate the Δt_{sens}, we have to compare the $t_s < fusionoutput >$ with the $t_s < radarmeas >$. However, the challenge here is that very few timestamps are known. For the rest of them we should be able to produce them. For this reason, the estimation of offsets among the clocks as well as the mapping of the timestamps from one local timeline to another are necessary tasks to create the mathematical model.

Since there is a lack of a common time origin of the clocks, offsets exist among them and we aim to define them mathematically.

We start by trying to find the offset between the C_s and C_f clocks. We use the

property that it was mentioned in Section 3.2.3 which give us the $t_s < radar\ meas >$ and the $t_f < radar\ meas >$, meaning that we have the timestamps for the same event from the two different processors. The offset can be found from the Equation 3.8:

$$\Delta(C_s, C_f) = t_f < radar\ meas > -t_s < radar\ meas > \tag{3.8}$$

To find the offset $\Delta(C_s, C_l)$ we should choose two events which are timestamped at the C_s clock and the C_l clock, and they need to refer to the same action. These events can be, for example, the output from a node at sensor processor and the receipt of this output at the logging system. The timestamps that satisfy the requirements are the $t_s < fusion\ output >$ and the $t_l < fusion\ output\ received\ at\ logger >$ and they are used for the calculation of the offset $\Delta(C_s, C_l)$ according to the equation:

$$\Delta(C_s, C_l) = t_l < fusion\ output\ received\ at\ logger > -t_s < fusion\ output > \tag{3.9}$$

As it can be seen in Figure 3.6 the $< fusion\ output >$ is sent as a UDP packet to the logger. However, before the UDP message sent there is a processing time to determine the packet's destination, prepare the header of the message etc. After that, it is placed in the UDP queue for its transmission. Then, it follows a transmission and a propagation delay while it is sending through the physical link. Finally, when it reaches the logger it is also queued and it waits to be read and written.

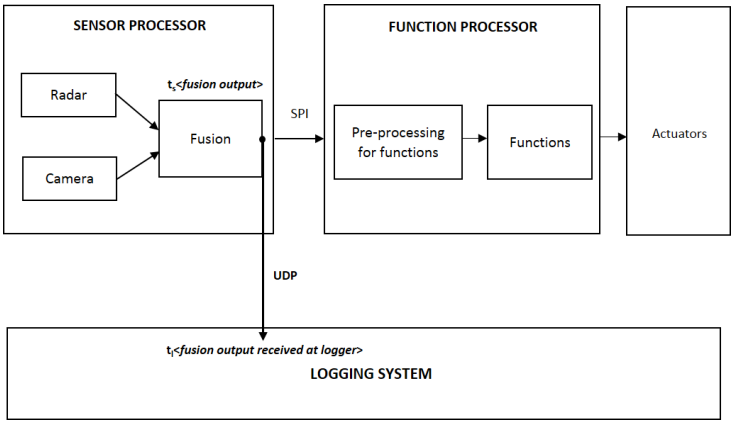


Figure 3.6. Logging of a timestamped event.

Those delays, known also as nodal delays, were explained in Section 3.1.1. As a result the complete equation which calculates the offset is given by:

$$\Delta(C_s, C_l) = t_l < fusion\ output\ received\ at\ logger > -t_s < fusion\ output > -d_{nodal} \tag{3.10}$$

where the d_{nodal} is given by:

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

and in the d_{queue} are included both the delay at transmission queue and in the receive queue.

The nodal delay depends on the load of the network. In addition, if we compare the $t_s < fusion\ output >$ and the $t_s < fusion\ output\ sent\ to\ logger >$, it is possible to calculate the d_{proc}. After calculations of this delay in our system, the d_{proc} results to be less than $200\mu s$. Since this delay is in timescale of μs and it is not possible to calculate the rest of the nodal delays, because the logging system and the sensor processor have separate clocks which are not synchronized, for the present thesis we assume that the nodal delay will be zero.

$$d_{nodal} \simeq 0 \tag{3.11}$$

The offset $\Delta(C_f, C_l)$ can be calculated by the Equation 3.7. We make the same assumption as before when we calculate this offset.

We now wish to estimate the time t_i on the function processor that is associated with an external event e_i occurred on sensor. For example, we want to know the time t_f that is given from P_f that the $< camera\ meas >$ occurred. Having estimated the offset $\Delta(C_s, C_f)$, our request can be implemented and all the timestamps that exist only in the sensor timeline can be mapped to the function timeline.

$$t_f < camera\ meas >= t_s < camera\ meas > +\Delta(C_s, C_f) \tag{3.12}$$

$$t_f < fusion\ output >= t_s < fusion\ output > +\Delta(C_s, C_f) \tag{3.13}$$

Accordingly, the timestamps that exist only in function timeline can be mapped in the sensor timeline.

$$t_s < function\ input >= t_f < function\ input > -\Delta(C_s, C_f) \tag{3.14}$$

It is also possible to map all the timestamps that exist in the function timeline to the logger timeline and the opposite.

$$t_l < functions\ input >= t_f < functions\ input > +\Delta(C_f, C_l) \tag{3.15}$$

$$t_f < functions\ output\ received\ at\ logger > = t_l < functions\ output\ received\ at\ logger > - \Delta(C_f, C_l) \tag{3.16}$$

4 METHODS

This Chapter describes the methods used to calculate the delays.

We have modelled the use-case system and we will continue with calculating the delays which were defined in the Section 3.2.1 and shown in Figure 3.4. The delays are summarized below:

- Δt_{sens} is the processing time between the radar measurement and the sensor fusion's output.
- Δt_{trans} is the time it takes to transfer the data trough the SPI as well as the time in the pre-processing for functions module.
- $\Delta t_{sens+trans}$ is the sum of the first two measurements, so from the radar measurement to the output of the pre-processing module.
- Δt_{funct} is the processing time in the functions.

The delays can be calculated by comparing the timestamps of different events. Since we do not have the possibility to interfere in the synchronization method among the nodes we will calculate all the delays based on the three different timelines by using statistical methods. The measurements will be compared to verify if the results are similar.

4.1 Methods

Statistical methods are developed to estimate experimentally the processing times of the tasks. The first method is based on the real measurement timestamps of the sensor, the second is based on the arrival time at the acquisition framework at the logger and the third is based on the measurements from the reference sensor.

To estimate the processing times we use statistical methods and we take advantage of the fact that there are different clocks in the system. We will estimate the times first based on the timestamped events from the sensor and function timelines where synchronization is applied and secondly calculate the same times based on the arrival time at acquisition framework at the logger and after that we compare the distributions of the calculations. If there are uncertainties we use a third method where the delays are calculated based on the reference data that we have from the reference system.

4.1.1 Estimation of delays based on sensor's and function's timelines

In this section we are going to compare events that are timestamped only at C_s or C_f.

- Δt_{sens}

This delay includes the time that it takes for the sensors to acquire their raw data and their processing from the sensor fusion. The delay is estimated from the Equation 4.1 by subtracting the timestamps of $< radar\ meas >$ and $< fusion\ output >$ events.

$$\Delta t_{sens} = t_s < fusion\ output > - t_s < radar\ meas > \tag{4.1}$$

Both of them are reported based on C_s clock and the comparison between them does not include uncertainties.

- Δt_{trans}

This delay is calculated from the Equation 4.2

$$\Delta t_{trans} = t_f < functions\ input > - t_f < fusion\ output > \tag{4.2}$$

where $t_f < fusion\ output >$ is calculated from the Equation 3.13.

- $\Delta t_{sens+trans}$

This is the delay from when the radar sensor takes its measurement to when the pre-processing is completed on the function processor. After the pre-processing the data will be processed from the functions' algorithms in order to decide for the active safety system's reaction. In this point it is important that the functions have updated data compensated for the time delay of the $\Delta t_{sens+trans}$ to make right decisions. This time is very critical for the system's good performance. This time is the sum of the two previous calculated delays according to Equation 4.3

$$\Delta t_{sens+trans} = \Delta t_{sens} + \Delta t_{trans} \tag{4.3}$$

or in terms of timestamped events the delay is given from the Equation 4.4

$$\Delta t_{sens+trans} = t_f < functions\ input > - t_f < radar\ meas > \tag{4.4}$$

4.1.2 Estimation of delays based on logger's timelines

During this method, we are going to compare events that timestamped only from C_l. To distinguish the estimation of delays with the second method from the first, the delays in this Section are written with *capital T*.

- ΔT_{sens}

Before we extract the equation to calculate the ΔT_{sens} delay, it is necessary to insert two new notations. Those are the $t_{lcalc} < radar\ meas\ received\ at\ logger >$ and the $t_{lcalc} < fusion\ output\ received\ at\ logger >$.

After the radar sensor has taken its measurement, this information will be sent to the logger and this event will be timestamped with $t_s < radar\ meas\ sent\ to\ logger >$. However, we do not know when this measurement will be sent. It can happen immediately after the measurement is taken or much later (see Figure 4.1). What we need to know to calculate the ΔT_{sens} is the logger's time when the radar measurement would be sent to the logger immediately after the measurement. This is calculated by the Equation 4.5.

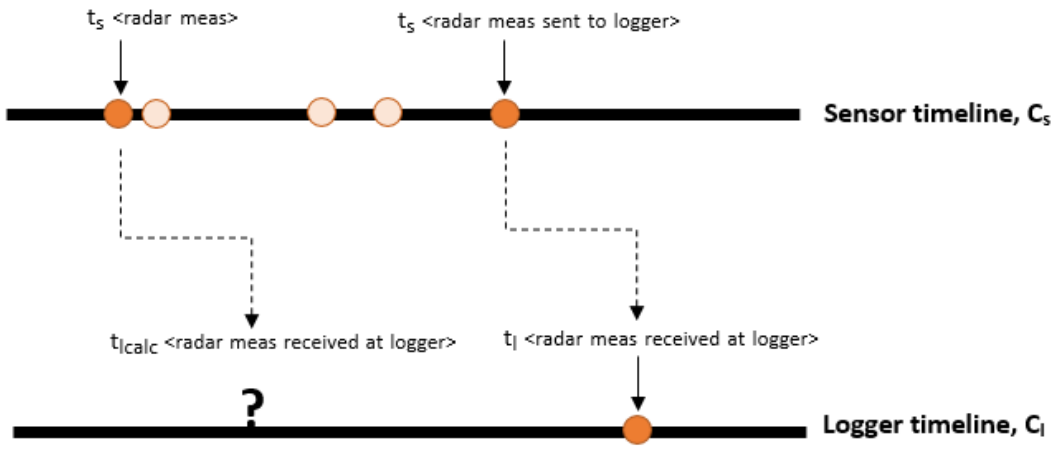


Figure 4.1. Illustration of events timestamped from sensor and logger timeline.

$$t_{lcalc} < radar\ meas\ received\ at\ logger > = t_l < radar\ meas\ received\ at\ logger > -(t_s < radar\ meas\ sent\ to\ logger > -t_s < radar\ meas >) \tag{4.5}$$

Exactly the same applies for the fusion output. The Equation which gives the $t_{lcalc} < fusion\ output\ received\ at\ logger >$ is the 4.6.

As a result, the calculation of the ΔT_{sens} is not based on pure logger's time. However, we aim to calculate this time and compare the results with the same calculation Δt_{sens} from the first method to see how close the distributions are.

$$t_{lcalc} < fusion\ output\ received\ at\ logger > = t_l < fusion\ output\ received\ at\ logger > \\ -(t_s < fusion\ output\ sent\ to\ logger > - t_s < fusion\ output >) \tag{4.6}$$

Finally, the equation to calculate the delay ΔT_{sens} is given by 4.7.

$$\Delta T_{sens} = t_{lcalc} < fusion\ output\ received\ at\ logger > \\ - t_{lcalc} < radar\ meas\ received\ at\ logger > \tag{4.7}$$

- ΔT_{trans}

The ΔT_{trans} is given from the Equation 4.8.

$$\Delta T_{trans} = t_l < functions\ output\ received\ at\ logger > \\ - t_lcalc < fusion\ output\ received\ at\ logger > -\Delta T_{funct} \tag{4.8}$$

- ΔT_{funct}

This delay is the processing time on the functions node. Unfortunately, there are no events in the beginning and in the end of this node that are timestamped with the same clock. Instead, there is the $t_f < functions\ input >$ in the function's timeline and the $t_l < functions\ output\ received\ at\ logger >$ in logger's timeline. To calculate the delay we will try to compute the time that the $< functions\ input >$ would be theoretically received at the logger if it was logged.

This is possible because there is some data that come immediately from the backbone of the car and they are logged very often through FlexRay at the logger. The same information comes through the sensor, they are compared to $t_f < functions\ input >$ and go through the functions node as all the other information will go. The node takes the last available information from the backbone, just the one before the $t_f < functions\ input >$. So the information is delayed until the processing in this node is finished and then it is logged. As a result, the same information is written at the logger in the beginning of functions node and in the end of it. If we compare those two times the time difference will give us the Δt_{funct}.

Since the node takes the last available information from the backbone and let us assume that this comes every $x$ms, the worst case scenario will be to exist an uncertainty of $x$ms for the Δt_{funct}.

4.1.3 Estimation of delays based on reference data

For the estimation of the delays with this method, only reference data are used.

To distinguish this method's delays from the other methods we will use the notation of τ, for example $\Delta\tau_{trans}$. With this method we estimate only the $\Delta\tau_{trans}$ because it seems that this delay is very dynamic and it is good to have a third chance of distribution estimation.

- $\Delta\tau_{trans}$

When the data is received at the logger it is timestamped with the logger's clock. This is applied for all the data that they are logged through all the interfaces, for example data that are logged through Ethernet, CAN bus etc. Their arrival time on the acquisition frame at the logger is based on the C_l clock. This helps in synchronization of multiple sensors.

This gives us the opportunity to synchronize the reference data from the reference system to the sensor data.

Initially we want to synchronize the reference data to the $t_l < fusion\ output\ received\ at\ logger >$. This means that we will have reference data about the state estimates of the host and the target for all the timestamps when the fusion output is received at the logger. Then, we calculate the TTC (Time To Collision) for every sample in a logging file based on the reference data. The TTC is given by the Equation 4.9.

$$TTC = \frac{longitudinal\ position}{host\ velocity} \tag{4.9}$$

Then, we synchronize the data from the reference system to the $t_l < functions\ output\ received\ at\ logger >$ and we calculate again the TTC. For the last calculation we subtract the Δt_{funct} from TTC, in order to estimate the TTC when the functions input would be logged at the logger.

The time difference between the TTCs wil give the $\Delta\tau_{trans}$.

5 RESULTS AND DISCUSSION

In this chapter the results from the thesis are presented and are discussed in parallel. The above mentioned methods are applied to a large number of files which include the logged signals that are used to compute the statistics of the processing times.

5.1 Datasets

To examine how the different traffic situations can influence the delays tests were created which were conducted in a test track in Gothenburg and they include one car, the host, in which we log the data and it moves with constant speed and a standstill car in front of it, named as the target. The same scenario is repeated but now the host variates its speed. Finally, we use logging files which were collected from common drivings in different kind of roads and in cities with low and very high traffic. All the methods are applied in all the three data sets.

The logging files are categorized in three data sets which are summarized below:

- A set of logs with one target and constant host speed.
- A set of logs with one target and variable host speed.
- A set of logs with a lot of targets and variable host speed- dynamic driving.

The equations from the Chapter 4 of each method are applied to the three data sets and the results are presented below.

All the results are normalized to the maximum delay which has been calculated. This results in a maximum delay which is equal to 1 and all the other delays are adjusted proportionally.

5.2 Results

We succeeded first to calculate the offsets among the clocks based on the Equations 3.7, 3.8 and 3.9. The offsets will be different for every logging file because of the different notion of time from the clocks. For example, the sensor's clock might be reset to zero after a specific period of time while the logger might measure the time continuously from the moment when the car is turned on.

For a specific logging file, which is picked up randomly, the results for the different offsets are shown at Figures 5.1, 5.2, and 5.3. Those offsets are valid only for the specific logging file and they are not applicable to all of them.

The results for the offsets $\Delta(C_s, C_l)$ and $\Delta(C_f, C_l)$ include a big variation since it is among the two processors and the logger, which include the delays over the Ethernet.

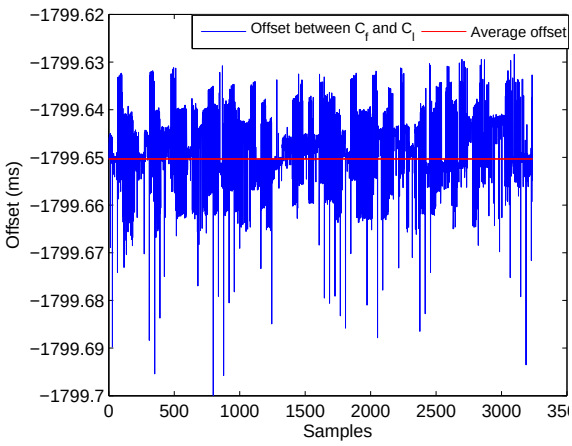


Figure 5.1. The offset between C_f and C_l

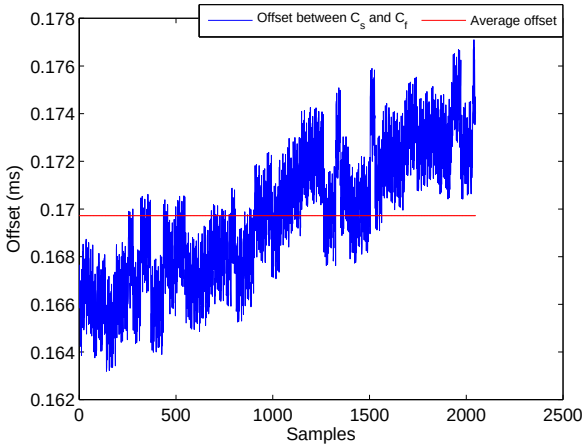


Figure 5.2. The offset between C_s and C_f

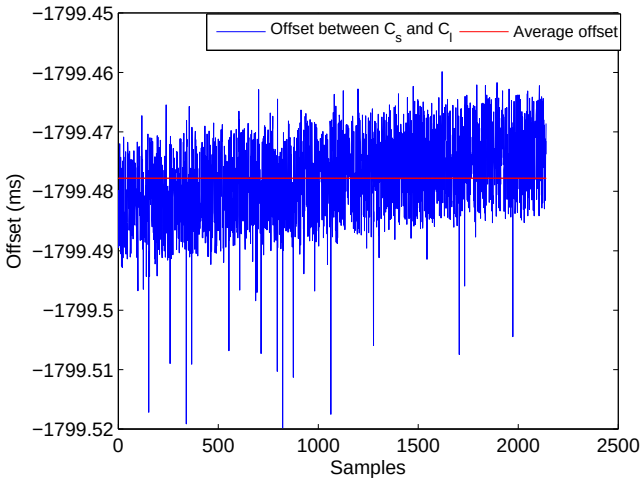


Figure 5.3. The offset between C_s and C_l.

Although different offsets exist, it is still possible to calculate the drift between the sensor and processor clocks where synchronization is applied. The drift for the specific logging file can be seen at Figure 5.2. Further calculations are made on a large number of logging files in order to find the average drift which is estimated to be $0.015ms$ for one minute log, a number also normalized to the maximum delay.

From now and on, the statistical results from the computation of the delays after we apply the three different methods to the three different data sets are given. The methods are applied to around 90 logging files with 2 hours total duration containing data from driving on public roads and at specific test tracks.

In the first method the events which are timestamped either from C_s or from C_f clocks are compared and the numerical values of the delays Δt_{sens}, Δt_{trans} and $\Delta t_{sens+trans}$ are presented. In the second method events which are timestamped only from the logger's clock C_l are compared and the numerical results of the delays ΔT_{sens}, ΔT_{trans} and ΔT_{funct} are given. In the third method the delay of $\Delta \tau_{trans}$ is given based on data from the reference sensor.

a. **Estimation of delays based on sensor's and function's timelines**

The delays Δt_{sens}, Δt_{trans} and $\Delta t_{sens+trans}$ have been calculated respectively from the Equations 4.1, 4.2 and 4.3. The same delays are calculated for the logs which include data with constant speed, variable speed, dynamic driving as well as for the summation of them. Statistical results from this first method are presented in the Table 5.1.

Table 5.1. Statistical results from the first method

	Constant speed			Variable speed			Dynamic driving		
	$\Delta t_{sens+trans}$	Δt_{sens}	Δt_{trans}	$\Delta t_{sens+trans}$	Δt_{sens}	Δt_{trans}	$\Delta t_{sens+trans}$	Δt_{sens}	Δt_{trans}
Mean	0.563	0.310	0.254	0.602	0.360	0.243	0.593	0.349	0.244
Sigma	0.069	0.079	0.060	0.073	0.070	0.054	0.072	0.071	0.054
Median	0.572	0.360	0.249	0.606	0.381	0.241	0.597	0.379	0.241
Samples	39000			7500			60000		

We notice that there is a difference in the measurements of the dispersion of the delays when the host moves with constant speed and when it moves with variable or it drives more dynamically.

By looking the $\Delta t_{sens+trans}$ delay,which is the processing time from the radar measurement to the output of the pre-processing module, we notice that this is lower when the host drives with constant speed. In Figures 5.4, 5.5 and 5.6 we can see the box plots of the $\Delta t_{sens+trans}$ for the three different data sets. Box plot is a standardized way of displaying the distribution of the data based on the five number summary: minimum, first quartile, median, third quartile, and maximum. The central blue rectangle spans the first quartile to the third quartile. The segment inside the rectangle shows the median while the "whiskers" above and below the box show

the locations of the minimum and maximum. The surprisingly high maximums or surprisingly low minimums called outliers and are shown with the red crosses.

A bigger variation in distribution exists in the last two cases, in the variable speed and in the traffic. In the case of constant speed the mean value is $0.563ms$, in the case of variable speed it is $0.602ms$ while in traffic it is $0.593ms$.

The plot which includes the results from the overall logging files is given at Figure 5.7 where we can clearly recognise when the speed is not constant by the variation in the distributions and the more extreme values. In constant speed the full range of variation from minimum to maximum is distributed in similar way while in the other case the distribution is more outspread. The overall statistical results for the $\Delta t_{sens+trans}$ calculated by the first method which was applied to the summation of the logging files are given analytically in the Table 5.4.

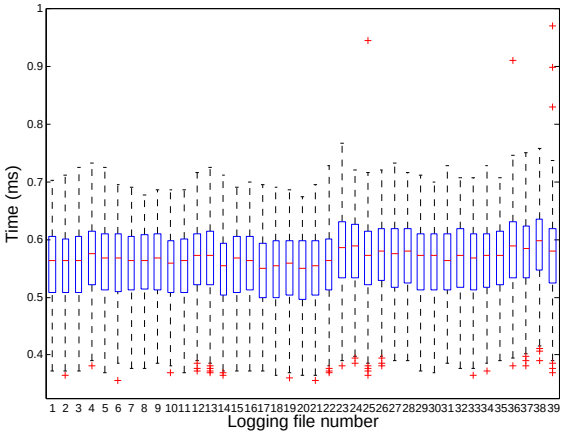


Figure 5.4. Boxplot of $\Delta t_{sens+trans}$ with constant speed

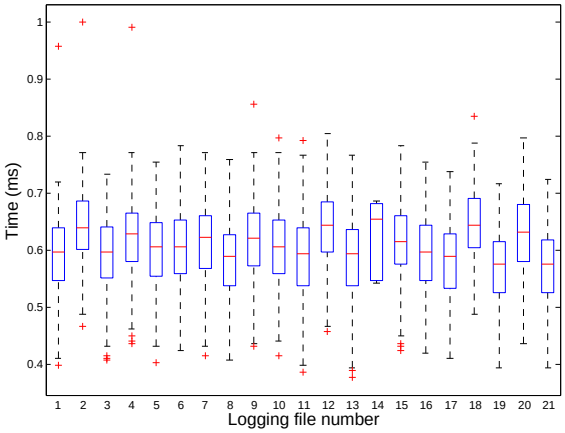


Figure 5.5. Boxplot of $\Delta t_{sens+trans}$ with variable speed

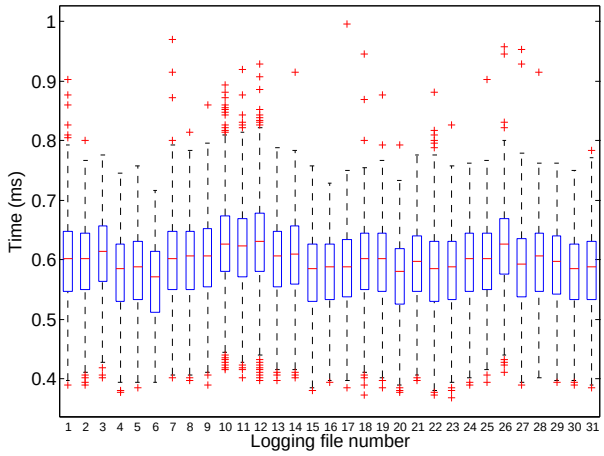


Figure 5.6. Boxplot of $\Delta t_{sens+trans}$ in traffic.

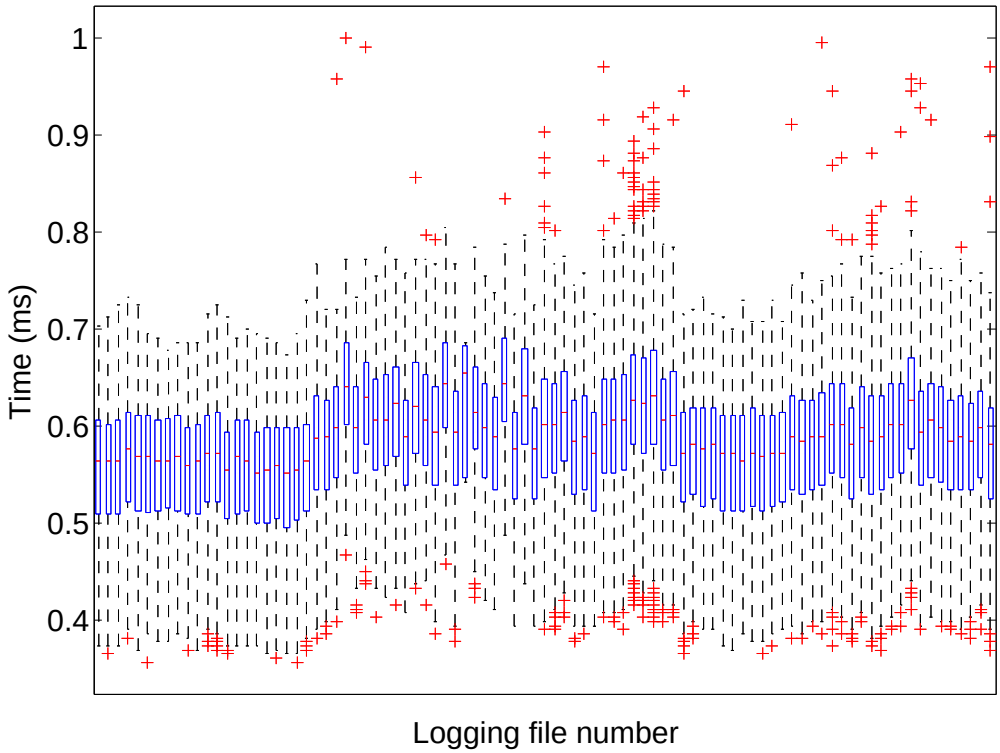


Figure 5.7. Boxplot of $\Delta t_{sens+trans}$ of all logging files.

By examining the results for the Δt_{sens}, which is the processing time that the sensors make their measurements and the data is fused, from the Table 5.1 we notice that there is a similar analogy in these results with the $\Delta t_{sens+trans}$ results which means that the delay is lower for the data set of the constant speed. The difference in the distribution of Δt_{sens} among the three data sets is shown in the Figure 5.8.

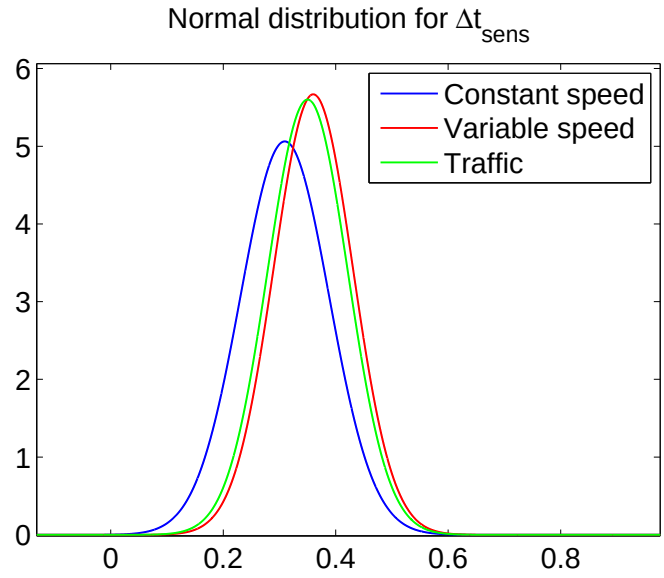


Figure 5.8. Distributions of the Δt_{sens} for the three data sets.

Examining the results for the Δt_{trans}, which is the processing time for transferring the data over the SPI and pre-process them to be ready for the functions, it is noticed that there is a better symphony among the three data sets for this delay which can be seen from the comparison of the normal distributions for the data sets in the Figure 5.9.

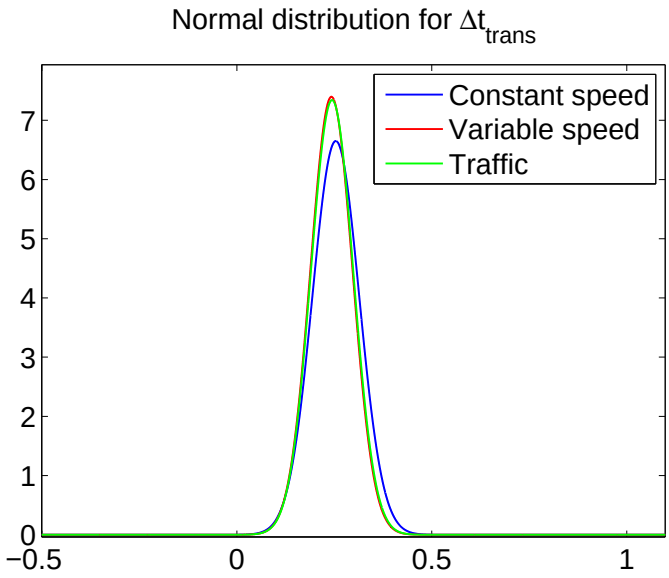


Figure 5.9. Distributions of the Δt_{trans} for the three data sets.

As a result, it seems that the Δt_{sens} delay depends on the complexity of the scenario, which means that it varies a lot depending on the different speeds and the amount of targets that are detected from the sensor. The more objects are detected, the bigger the delay will be. On the other hand, the Δt_{trans} seems to be not so dynamic as the Δt_{sens}.

Consequently, the variation in the $\Delta t_{sens+trans}$ delay which is shown in the Figure 5.7 is due to the delays during the detection and the fusion of the objects from the sensor and not due to the transferring of the data over the SPI.

A comparison between the Δt_{sens} and Δt_{trans} for the total amount of the logging files can be seen in the box plots of Figures 5.10 and 5.11 while analytical results for these delays can be found in the Table 5.4.

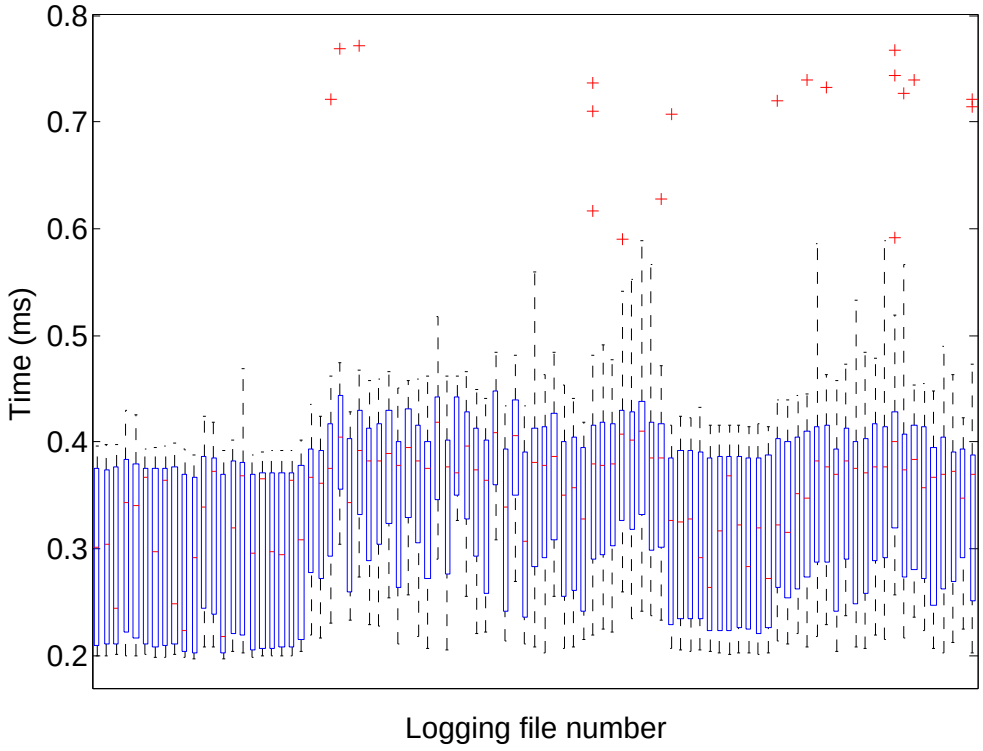


Figure 5.10. Boxplot of Δt_{sens} of all logging files.

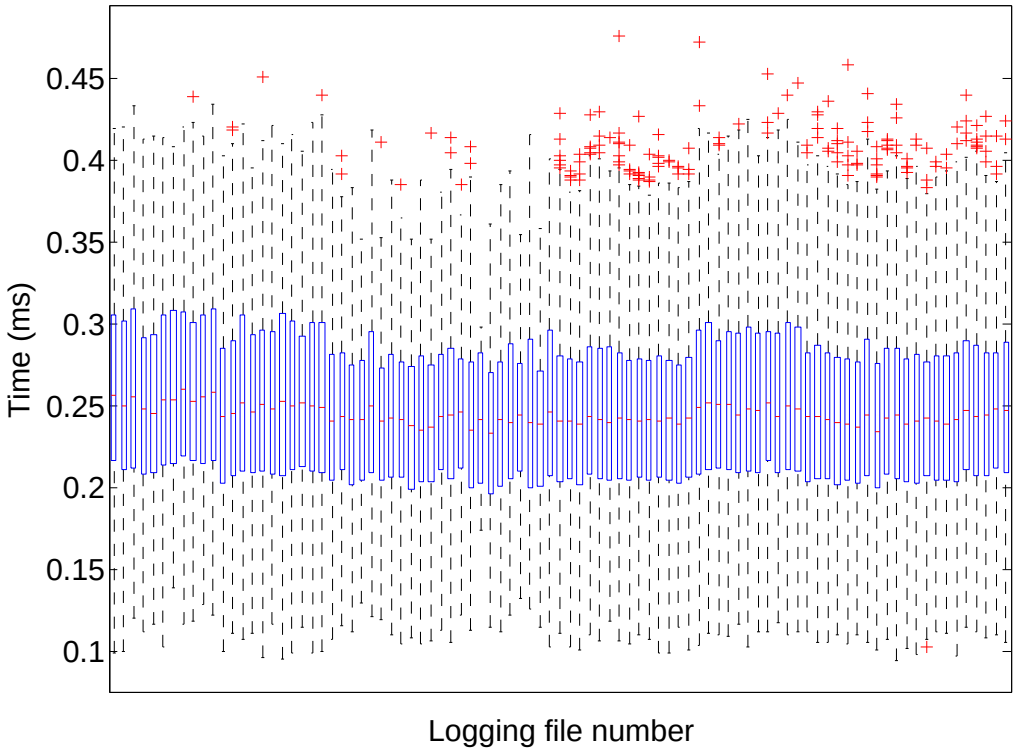


Figure 5.11. Boxplot of Δt_{trans} of all logging files.

b. **Estimation of delays based on logger's timelines**

The results of this Section are generated by using events which only timestamped from the logger's clock. The statistical results of ΔT_{sens} and the ΔT_{trans} for the three different data sets are presented at the Table 5.2.

Table 5.2. Statistical results from the second method

	Constant speed		Variable speed		Dynamic driving	
	ΔT_{sens}	ΔT_{trans}	ΔT_{sens}	ΔT_{trans}	ΔT_{sens}	ΔT_{trans}
Mean	0.311	0.251	0.361	0.238	0.350	0.252
Sigma	0.079	0.063	0.071	0.055	0.072	0.060
Median	0.356	0.252	0.380	0.233	0.375	0.253
Samples	39000		7500		60000	

Comparing the ΔT_{sens} among the data sets we notice that this delay is lower at the logs with the constant speed. As it was noticed from the calculations of Δt_{sens} with the first method, when the scenario is less complicated with very few obstacles

and constant speed this delay will be lower since the less objects are detected in the traffic environment the less time the fusion of them will take.

Regarding the ΔT_{trans} delay that it takes from the data to be sent over the SPI and pre-processed there is a closer agreement among the three data sets.

In addition, the delay of the ΔT_{funct} is estimated to be $0.106ms$. To calculate this delay we choose a signal that comes immediately from the backbone of the car and it is logged very often at the logger. The same signal comes through the sensor, it is compared to $t_f < functions\ input >$ and go through the functions node as all the other information will go. So, the same information is written at the logger in the beginning of functions node and in the end of it. The comparison between those two timestamps gives the ΔT_{funct} result of $0.106ms$.

The signal which was chosen was the ego vehicle longitudinal velocity which had the smallest sampling interval. A signal with very small sampling interval helps to avoid entering uncertainties in the calculations since the functions node takes the last available information from the backbone which is going through the functions and logged.

c. **Estimation of delays based on reference data**

While in the previous methods the timestamps from the events are used for the computation of the delays, in this method for the estimation of $\Delta\tau_{trans}$ they are not used timestamps but the state estimates from the reference sensor. Specifically, the longitudinal position and the host speed are used which give us how much time it will take for the host vehicle to collide with an obstacle on its path. This method is not applied to the previously defined data sets because there are not reference data for them. Instead, another data set is used where the signals have been logged from both the examined sensor and the reference sensor. The data set includes logging files where the host drives with constant speed towards a target which moves with lower speed than the host's.

Then, the reference data is synchronized first to the $t_l < fusion\ output\ received\ at\ logger >$ and later to the $t_l < function\ output\ received\ at\ logger >$ and the TTC is calculated for both cases. The difference between the TTCs give the $\Delta\tau_{trans}$ delay which is presented in the Table 5.3. For the same data set the Δt_{trans}, using the first method, and the ΔT_{trans}, from the second method, are calculated again and the results are also shown in the Table 5.3.

Comparing the results in the Table 5.3, we can conclude that the values for Δt_{trans} and ΔT_{trans}, calculated using different methods, agree with the value for $\Delta\tau_{trans}$ which was calculated using reference data.

Table 5.3. Statistical results from the reference data

	Constant speed		
	Δt_{trans}	ΔT_{trans}	$\Delta \tau_{trans}$
Mean	0.256	0.251	0.263
Sigma	0.062	0.063	0.094
Median	0.25	0.253	0.307
Samples	6500		

d. **Overall results**

The equations of the first and second method are applied to the summation of the logging files (around 90 logging files and 106500 samples) and the overall results are gathered and shown in the Table 5.4.

Table 5.4. Overall results

	First method			Second method	
	$\Delta t_{sens+trans}$	Δt_{sens}	Δt_{trans}	ΔT_{sens}	ΔT_{trans}
Mean	0.583	0.336	0.248	0.337	0.251
Sigma	0.073	0.077	0.057	0.077	0.061
Median	0.589	0.368	0.244	0.363	0.253
Samples	106500				

There is a close agreement between the delay Δt_{sens} and ΔT_{sens} where for the Δt_{sens} the mean value is 0.336 with sigma 0.077 while for the ΔT_{sens} the mean value is 0.337 with sigma at 0.077. As a result, there is an accordance in the two methods estimating the delay from the sensors' raw data acquisition to the completion of fusion at the sensor processor. According to the Table 5.4, a similar accordance exists between the Δt_{trans} and ΔT_{trans}.

Therefore, the calculations of the processing times based on the two methods give very similar results, which results in a good system performance.

6 CONCLUSION AND FUTURE WORK

This Chapter summarizes briefly the problem, the methods and the results as well as it suggests ways on the next steps for the continuation of the thesis project.

The thesis addresses the problem and the question if there is or not high accuracy of the correlation of the signals, which are produced by a sensor network, to each other and to the real traffic situations. The limited accuracy is caused by the computationally heavy signal processing on the different nodes that the system consists of, like the sensors, the node where fusion is run, the node which pre-processes the data to be used as input to the node where the decision making is made. It is also caused by the transmission of the signals through different interfaces (Ethernet, CAN bus, SPI etc.). This problem reduces the performance of the active safety systems as well as it poses problems for their verification during the development stage. A real active safety system is used as a use-case in the thesis. This consists of three processors, the sensor processor, the function processor and the logging system with different physical clock each. Different events take place in the signal chain from an object detection from the sensors, to fusion and decision making. Each of those events are timestamped from the different clocks.

Since one of the processors is developed with a collaboration with an external supplier, there is a limitation to make changes and interfere to that hardware. As a result, there is no possibility to change the synchronization method between the two processors but it is assumed that it works accurately, although it can be a possible source of delays. To remove the uncertainties and to verify the accuracy of the active safety system different methods are developed to estimate statistically the delays that are caused by the heavy signal processing and the transmission of the data over different interfaces. The delays that are calculated are:

- the processing time between the radar measurement and the sensor fusions output
- the time it takes to transfer the data through the SPI as well as the time in the pre-processing for functions module
- the sum of the first two measurements, so from the radar measurement to the output of the pre-processing module
- the processing time in the functions

The first method is based on the sensors' and functions' processors timebase while the second on the logger' s timebase. Additionally, data coming from a reference sensor system is used to measure the same delays. The methods are applied to a large variety of collected data from drivings in different types of roads and conditions.

The overall results indicate that the delays depend on the complexity of the traffic environment. The more complicated the traffic conditions are, the higher delay the system has. What is more, the measurements of the delays from the different methods conclude to similar results and the outcome from the reference sensor agree with the outcome from the examined sensor system which results in a good system performance.

The limitation of the thesis project was the limited access to the architecture of the system and more specifically to the synchronization methods. The logger was not synchronized with the other two processors and the verification of the synchronization of the sensors and functions processors was not an easy task.

What it is proposed for continuation of this thesis work is synchronization of the logger with the other processors. If it was synchronized we would know the exact time when an event occurs with the logger's clock without this time to be exposed in UDP delays or in buffer or write delays.

Another suggestion for further development of the system, and especially if it is going to be used in combination with a lot more sensors, is to set the GPS time in the logger. Using a reference system then, as we did in the third method, we will have the accuracy of the examined system in the timescale of *ns*.

Bibliography

Brahmi, M., Schueler, K., Bouzouraa, S., Maurer, M., Siedersberger, K. and Hofmann, U. (2013). Timestamping and latency analysis for multi-sensor perception systems. *SENSORS, 2013 IEEE*.

Flaviu, C. (1989). Probabilistic clock synchronization. *Distributed Computing*, pp 146–158.

Hazra, R., Bhattacharyya, D., Dey, S., Feruza, S. and Furkhat, T. (2009). Network issues in clock synchronization on distributed database. *International Journal of Database Theory and Application*.

Heidemann, J. and Govindan, R. (2005). Embedded sensor networks. In: *Handbook of networked and embedded control systems*, W. Levine D. Hristu-Varsakelis (Ed.). Birkhuser. Boston.

Huck, T., Westenberger, A., Fritzsche, M., Schwarz, T. and Dietmayer, K. (2011). Precise timestamping and temporal synchronization in multi-sensor fusion. *2011 IEEE Intelligent Vehicles Symposium*.

Koopman, P. (1996). Embedded system design issues (the rest of the story). *Proceedings of the International Conference on Computer Design (ICCD 96)*.

Kopetz, H. and Ochsenreiter, W. (1987). Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, **36**(8), 933–940.

Kurose, J. and Ross, K. (Eds.) (2013). *Computer Networking - A Top-Down Approach*. Addison Wesley Longman.

Lamport, L. (1987). Time, clocks, and the ordering of events in a distributed system. *Magazine Communications of the ACM*, **21**, 558–565.

Mills, D. L. (1991). Internet time synchronization: The network time protocol. *IEEE Transactions on Communications*.

Olson, E. and Arbor, A. (2010). A passive solution to the sensor synchronization problem. *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, **28**, 1059 – 1064.

Rhee, I., Lee, J., Kim, J., Serpedin, E. and Wu, Y. (2009). Clock synchronization in wireless sensor networks: An overview. *Sensors 2009*, pp 56–85.

Sivrikaya, F. and Yener, B. (2004). Time synchronization in sensor networks: A survey. *IEEE Network*.

Stojmenovic, I. (Ed.) (2005). *Handbook of sensor networks: algorithms and architectures*. John Wiley 'I&' Sons. Hoboken, New Jersey.

Sundararaman, B., Buy, U. and Kshemkalyani, A. D. (2005). Clock synchronization in wireless sensor networks: a survey. *Ad Hoc Networks 3*, pp 281–323.

Westenberger, A., Daimler, A., Huck, T., Fritzsche and Schwarz, T. (2011). Temporal synchronization in multi-sensor fusion for future driver assistance systems. *Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2011 International IEEE Symposium*.

Zhao, F. and Guibas, L. (2004). *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann. San Francisco, CA, USA.