Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Roman Trüb

Low-Error Multi-Hop Clock Synchronization

Semester Thesis SA-2015-20
14 September to 18 December 2015

Supervisors: Roman Lim
Dr. Olga Saukh
Professor: Prof. Dr. Lothar Thiele

Acknowledgements

I would like to thank my project advisors Roman Lim and Dr. Olga Saukh for their help and support during the semester thesis. Furthermore I would like to thank Prof. Dr. Lothar Thiele for giving me the opportunity to write this semester thesis in his research group.

Abstract

Time synchronization is an important basic service for wireless sensor networks (WSNs). A lot of time synchronization protocols for WSNs use linear regression to propagate the reference time from hop to hop through the network. State-of-the-art synchronization algorithms use the ordinary least-squares (OLS) regression. Because OLS has properties that are not ideal in the context of multi-hop time synchronization alternative linear regression algorithms promise better accuracy.

In this semester thesis the use of an alternative linear regression, the geometric mean regression (GMR), in conjunction with the widely used FTSP time synchronization protocol for WSNs is examined. This thesis presents a performance analysis of the FTSP protocol on a 27 hop testbed as well as an evaluation when the use of GMR is beneficial compared to the current implementation.

Contents

Chapter 1

Introduction

1.1 Motivation

Time synchronization is an essential basic service in wireless sensor networks (WSNs). It is used to perform coordinated actions (e.g. simultaneously measure the environment) at different nodes or to obtain consistent data (e.g. order events recorded over the whole network chronologically). An example application for which an accurate time synchronization is crucial is the localization of a shooter by a WSN [10]. The location can be calculated from the measured arrival times of the acoustic event at different sensor positions.

The clocks of the nodes cannot be perfectly manufactured and therefore exhibit a drift. To compensate this drift a synchronization algorithm is required. Usually such algorithms work in a hop-by-hop manner as depicted in Figure 1.1. This means that a reference node first synchronizes the nodes in its vicinity (e.g. limited by radio range). Once these nodes are synchronized they can themselves synchronize the nodes in their vicinity. This goes on until all the nodes in the network are synchronized to the reference node.

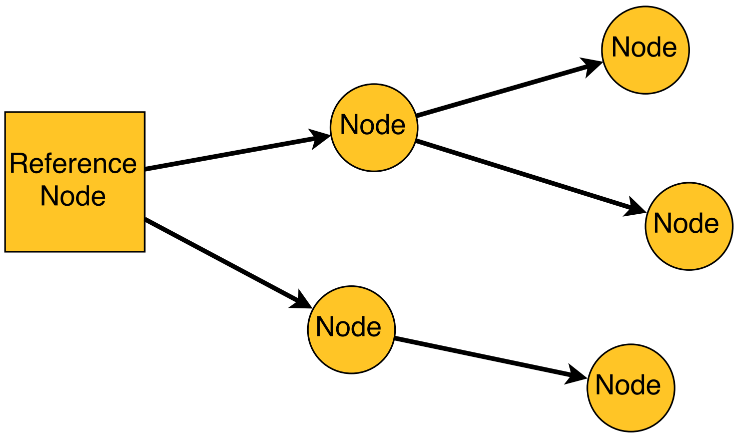


Figure 1.1: Hop-by-hop time synchronization in a WSN

Because the nodes are not always perfectly synchronized to the previous nodes every node introduces a small error when the reference time is propagated through the network. This error accumulates with multiple hops [5].

State-of-the-art time synchronization protocols for WSNs (e.g. FTSP [8]) use the ordinary least squares (OLS) regression as a linear regression to map their local time to the synchronized global time of the network. Unfortunately OLS exhibits some properties that aren't ideal in the context of multi-hop time synchronization. Most importantly the slope calculated by OLS exhibits a bias towards zero. A promising alternative linear regression algorithm is the geometric mean regression (GMR). In contrast to OLS it exhibits significantly less bias towards zero.

In this thesis we first analyse OLS regression formally in order to understand which parameters influence the bias and how strong their influence is. From this we formulate hypotheses which describe the relations. Then we conduct tests on a real WSN using a testbed. With the obtained results we check whether the expectations are met. Because the testbed has hardware limitations we also use a simulation to investigate the relations.

1.2 Goal

The goal of this semester thesis is to investigate the performance of the FTSP protocol and its linear regression implementation. The performance analysis is based on traces obtained by running FTSP on a testbed as well as by simulating the time synchronization protocol. Furthermore the use of alternative linear regressions in FTSP shall be evaluated. If an alternative linear regression promises time synchronization with better accuracy it shall be implemented and evaluated on the testbed.

1.3 Findings

In this thesis we derive a formula to estimate the slope which suffers from a bias caused by the OLS regression. With tests on a testbed as well as simulations we can confirm all but one hypotheses we formulated based on the formula. In addition the analysis of the data from the testbed shows when running FTSP on real WSN the number of fresh synchronization updates decreases with the number of hops. Furthermore we show that sensor nodes exhibit oscillations over time which depend on the number of regression table entries.

1.4 Outline

The remainder of this thesis is structured as follows. Chapter 2 references related work, highlights the key properties of the used linear regression algorithms and explains FTSP. Chapter 3 lists the evaluation metrics and formally analyses the OLS regression. Chapter 4 describes the process of obtaining real data from a testbed and presents the corresponding results. In Chapter 5 we explain the simulation and present the results. Chapter 6 concludes the thesis and discusses future work.

Chapter 2

Background

This chapter provides background information about the topics discussed in the following chapters. First we reference related work. Then the important properties of the two linear regressions examined in this thesis are discussed. Finally the FTSP protocol is explained.

2.1 Related Work

In this thesis the flooding time synchronization protocol (FTSP) protocol which is described in [8] is used. An explanation which highlights the important aspects of this time synchronization protocol is given in section 2.3.

Another important work related to this thesis is the paper about reducing multi-hop calibration errors from Saukh et al. [9]. In this paper it is shown that GMR outperforms OLS when it is used to calibrate multi-hop sensor networks.

2.2 Linear Regression

2.2.1 Ordinary Least Squares Regression

The ordinary least squares (OLS) regression is a widely used method for line fitting. It minimizes the sum of the vertical distances (resdiuals) between the regression line and the data points depicted in Figure 2.1.

The OLS regression is not symmetric, i.e. there is a independent variable (x-axis) and a dependent variable (y-axis) which cannot be swapped in order to get the same result. The non-symmetry property is observable in Figure 2.3. The line labeled *OLS* regresses the local (independent variable) on the global time (dependent variable) whereas the line labeled *OLS Swapped* regresses the global (dependent variable) on the local (independent variable) time.

Another important property is that OLS regression exhibits a bias towards zero in the computed slope. This bias depends on the noise in the data. The problem is known as *regression attenuation* or *regression dilution* [3] [4]. This property is important in the context of error accumulation in multi-hop time synchronization since the bias increases with every hop.

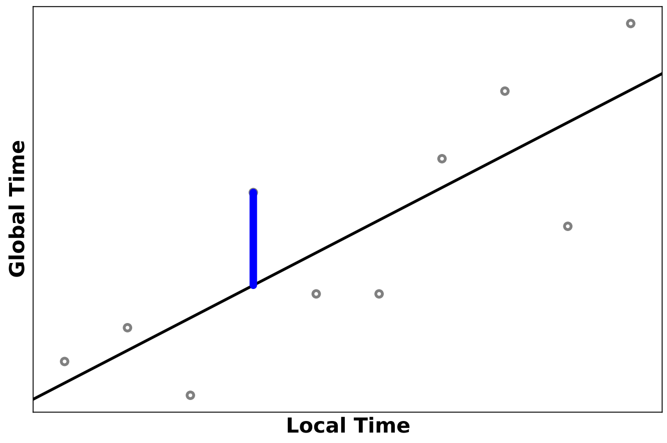


Figure 2.1: OLS minimizes the sum of the vertical distances

2.2.2 Geometric Mean Regression

The geometric mean regression (GMR) is a linear regression which minimizes the sum of the triangular areas spanned by the deviation of the data point from the regression line in both axis directions depicted in Figure 2.2. In contrast to OLS the GMR is symmetric and always lies between the two lines of the OLS regression. This is depicted in Figure 2.3. In addition it suffers less from a bias towards zero.

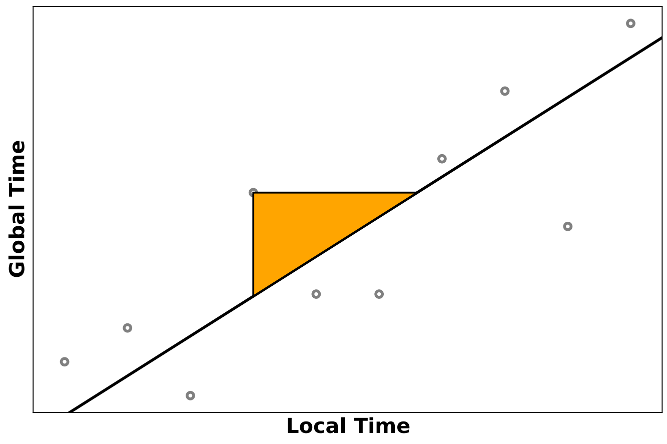


Figure 2.2: GMR minimizes the sum of triangles

Some properties of the two linear regressions are summaries in Table 2.1.

Table 2.1: Properties of linear regression algorithms

	OLS	**GMR**
Symmetry	no	yes
Scale-invariance	no	yes
Bias towards zero	yes	less severe

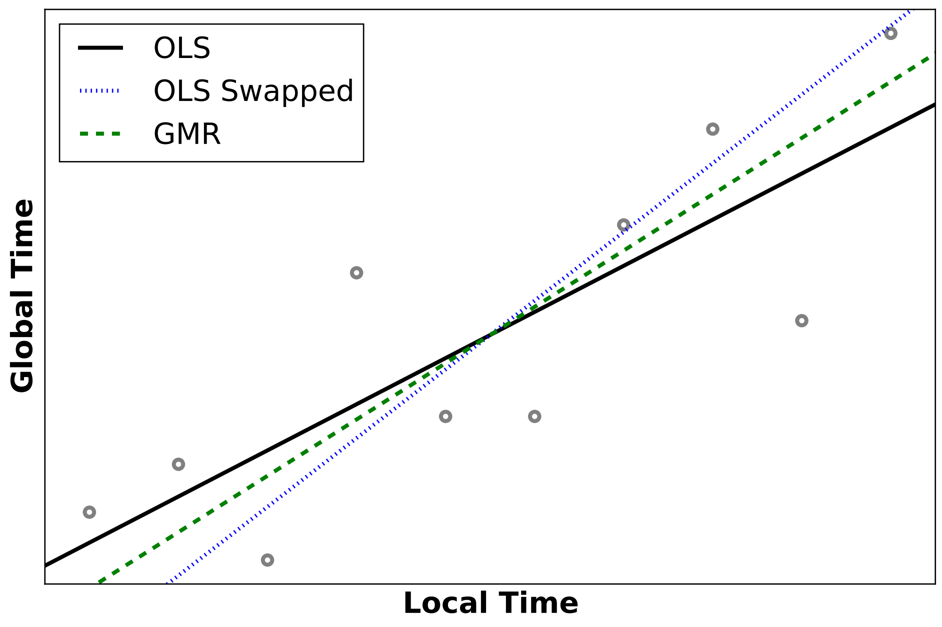


Figure 2.3: Comparison of OLS and GMR

2.3 FTSP

The Flooding Time Synchronization Protocol (FTSP) [8] is a popular time synchronization protocol for sensor networks. It is optimized for resource limited wireless platforms and and uses only low communication bandwidth. This makes it scalable. In addition it dynamically and implicitly updates the topology so it can handle node and link failures. In the following we describe the working principle.

2.3.1 General Working Principle

The root node acts as the reference. All other nodes synchronize their estimate of the global time to the time of the root node. The root node periodically floods synchronization messages throughout the network by sending broadcast messages.

The nodes synchronize their estimate of the global time at discrete points in time (with each synchronization message) to the global time. In order to reduce the communication overhead but yet to achieve high precision between the synchronization messages, FTSP compensates the clock drift by using linear regression. For this it is assumed that the offset between the clocks of two sensor nodes changes in a linear fashion over time. This is generally a good approximation if the short term stability of the clocks is good (which usually is true with sensor nodes).

The FTSP implementation reduces the influence of many error sources to a minimum, e.g. by time-stamping the packets at the MAC-layer. One error source it doesn't eliminate is the propagation time. This is not possible with a single message per synchronization update. However the influence of this error source is very small. The propagation time is less than one microsecond for ranges under 300 meters.

2.3.2 Synchronization Point Update

The synchronization messages contain the sender's estimate of the global time. At the receiver the message is time-stamped with the local time of the receiver. By neglecting the propagation time both timestamps correspond to the same point in time. Such a pair of corresponding global and local timestamps represents one *synchronization point*. The receiver collects such synchronization points in its regression table (by default the table holds 8 synchronization points). By using linear regression the node can then calculate the skew and offset. These parameters allows it to map the local time of the nodes clock to the global time of the network.

If a node receives too many synchronization points (by default 3) that are in disagreement with previous estimates of the global time it clears its regression table and starts collecting new synchronization points.

2.3.3 Multi-hop Synchronization

Once a node is synchronized it can synchronize other nodes. Whenever a node has enough entries in its regression table it periodically sends synchronization messages. Other nodes in the radio range can then collect synchronization points from the broadcasted messages.

For reasons of efficiency and robustness against topology changes every node needs a unique ID. In addition each synchronization message doesn't only contain the estimated global time of the sender but also includes the root ID as known by the sender and a sequence number. The sequence number is incremented by the root node every time it sends a synchronization message. This means each sequence number is propagated through the whole network.

Redundancy Prevention

The sequence number identifies the update round. It is used to prevent redundant information in the regression table. A node adds a synchronization point to its regression table only if the message which contains the synchronization point has a sequence number that is larger than the previously received sequence number. Because the messages are distributed using broadcasts it's possible that one node gets multiple messages from the same update round via different paths in the network. Since the nodes do not increment the sequence number all received messages contain the same sequence number. Therefore the node will only add the synchronization point to the regression table contained in the message which arrives first.

The reason for this is that it is advantageous for the regression to have synchronization points distributed over a long period of time. If the points used for the linear regression are very close to each other even small disturbances have a big influence. Because the size of the regression table is limited this redundancy prevention mechanism is important.

Root Election

At startup or generally when a node doesn't receive any synchronization message for some time (defined by ROOT_TIMOUT) it declares itself to be the root. A root node doesn't need to be synchronized and therefore starts sending synchronization messages immediately after becoming a root. Whenever a node receives a message with a smaller root ID it updates its variable of the current root ID. As a consequence all root nodes with higher IDs give up to be a root. In the end only one root, the node with the smallest ID, exists.

If the root node fails for some reason a new root is elected. This node keeps sending the global time based on the previously calculated parameters (skew and offset) in order to prevent jumps in the global time of the network.

Convergence

A network has converged if all nodes are synchronized to the global time. The network needs some time to converge after startup or node failures.

The ideal convergence time after startup t_{cs} can be determined by the parameters of the protocol. The following formula is given in [8] and only holds with the assumption that all links are reliable and no regression table is cleared during convergence phase.

$$t_{cs} = P \cdot (M + N \cdot R) \tag{2.1}$$

P :	Synchronization period [in secs]
M :	Time to declare itself the root if no message was received [in sync periods]
N :	Number of entries in the regression table needed to send synchronization messages
R :	Radius of the network (measured from the root node)

After startup all nodes first will timeout and then declare itself to be the root $(P \cdot M)$. Then every node needs to collect enough synchronization points to be able to send synchronization messages $(P \cdot N \cdot R)$. In these synchronization messages the root ID propagates from the root node to the edge of the network.

Implementation of OLS

For reasons of computing efficiency and numerical accuracy the implementation of FTSP uses integer division. In addition from the skew, which always is very close to 1.0, 1.0 is subtracted because a float around 0.0 can be stored with more precision compared to a float around 1.0.

Chapter 3

Formal Analysis

In this chapter we formally analyse the current implementation of the linear regression in FTSP. More precisely we mathematically describe the influence of the bias that is caused by OLS regression in a multi-hop scenario. We use this description to formulate hypotheses of the behavior we expect to observe when running the FTSP protocol on a WSN.

First we explain the metrics used in this thesis. Afterwards we derive the formula to calculate the skew influenced by the bias. In the end we formulate the hypotheses based on the derived formula.

3.1 Evaluation Metrics

To understand the metrics it is important to understand the model which is used to estimate the global time. So we start by describing this model. Afterwards we describe the metrics.

3.1.1 Model to Estimate the Global Time

The OLS regression is used to estimate the global time between two time synchronization updates. The synchronization points of the last synchronization messages represent the data points for the linear regression. The resulting line parameters, offset (intercept) γ and skew (slope) β, are used to map the local time of a node to the global time of the network.

$$t_{global} = t_{local} \cdot \beta + \gamma + e \tag{3.1}$$

e represents the error of the estimated global time. This error increases (accumulates) with the number of hops between the node which estimates the global time and the reference node.

3.1.2 Time Difference

The metric *time difference* in this thesis corresponds to the difference of the the estimated global time of a node in the line and the global time of the reference node.

$$\text{time\_difference} = \text{globalTime}_i - \text{globalTime}_{ref} \tag{3.2}$$

The variable i corresponds to the node ID of a node in the line. If the time difference is positive the node's estimate of the global time lies in the future, if the time difference is negative the node's estimate of the global time lies in the past.

3.1.3 Skew

The skew is the slope obtained from regressing the local timestamps (x-axis) in the regression table of a node in the line to the global timestamps of the reference node (y-axis). We use this metric to measure the bias caused by the linear regression algorithm.

3.1.4 RMSE

We use the root-mean-square error (RMSE) to evaluate the quality of the regression. The RMSE measures the difference of the estimated values and the actually observed values. These differences are also called residuals. The timestamps of the unsynchronized node correspond to the x-axis (x_i values) whereas the timestamps of the reference node correspond to the y-axis (y_i values). The RMSE is a measure of how well the linear regression matches the relation between the reference node and the unsynchronized node. For the RMSE we use the following definition:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2} \tag{3.3}$$

n is the number of timestamps and the function $f()$ represents the estimated linear relation obtained from the linear regression. With $f()$ the local timestamps can be mapped to global timestamps.

3.2 Bias Formula

In this section we present the derivation of the bias formula which we use to formulate the hypotheses. The formula estimates the influence of the bias on the slope (skew) calculated by OLS regression. The formula is based on Eq. (10) and Eq. (11) in [9]. First we state the relation of the bias and the skew for one hop. Then we extend the formula in order to use it for a multi-hop scenario.

3.2.1 One Hop

The relation of the bias α and the slope (skew) β calculated by the OLS linear regression is given by

$$\beta = \beta_* \cdot \alpha \tag{3.4}$$

β_* is the true slope which does not suffer from a bias. This can be expressed in the following way: $\beta^* = \beta \cdot \frac{1}{\alpha}$, where $\frac{1}{\alpha}$ is called the *compensation*.

According to Eq. (11) in [9] the bias is given as follows

$$\alpha = \frac{\sigma^2_{x_*}}{\sigma^2_{x_*} + \sigma^2_\eta} \tag{3.5}$$

where $\sigma^2_{x_*}$ denotes the variance of the true (noise free) but not observable measurements (i.e. timestamps) and σ^2_η denotes the variance of the noise of the measurements. This noise is caused by the clock and the time-stamping.

3.2.2 Multiple Hops

According to formula (10) in [9] the skew (slope) between the reference node and the k-th node can be calculated using the following expression:

$$\beta^{Ref\rightarrow k} = \beta^{Ref\rightarrow 1} \cdot \beta^{1\rightarrow 2} \cdot \beta^{2\rightarrow 3} \cdot \ldots \cdot \beta^{(k-1)\rightarrow k} \tag{3.6}$$
$$= (\hat{\beta}^{Ref\rightarrow 1} \cdot \alpha_1) \cdot (\hat{\beta}^{1\rightarrow 2} \cdot \alpha_2) \cdot (\hat{\beta}^{2\rightarrow 3} \cdot \alpha_3) \cdot \ldots \cdot (\hat{\beta}^{(k-1)\rightarrow k} \cdot \alpha_k) \tag{3.7}$$

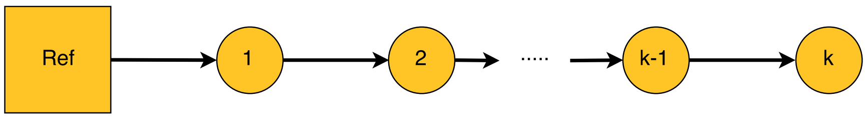


Figure 3.1: Numbering of the nodes

The numbering of the nodes is depicted in Figure 3.1. $\beta^{a\rightarrow b}$ means that the timestamps of node a is used as y-axis value and the timestamps of node b is used as x-axis value to compute the OLS regression. $\hat{\beta}^{(n-1)\rightarrow n}$ is an abbreviation for $\hat{\beta}^{(n-1)\rightarrow n} = \beta^{Ref\rightarrow 1} \cdot \beta^{1\rightarrow 2} \cdot \ldots \cdot \beta_*^{(n-1)\rightarrow n}$. It's possible to simplify the expression:

$$\beta^{Ref\rightarrow k} = (\beta_*^{Ref\rightarrow 1} \cdot \alpha_1)^k \cdot (\beta_*^{Ref\rightarrow 1} \cdot \alpha_2)^{k-1} \cdot \ldots \cdot (\beta_*^{(k-1)\rightarrow k} \cdot \alpha_k) \tag{3.8}$$

In the case of time synchronization all $\beta_*^{x\rightarrow y}$ $\forall x, y$ are very close to 1.0. In order to simplify the estimation of the bias the equality is assumed to hold. In addition the variance of the regression table entries as well as the variance of the noise are assumed to be constant for all nodes. With this the expression can be further simplified:

$$\beta^{Ref\rightarrow k} = \alpha^{\sum_{i=1}^{k} i} = \alpha^{\frac{k\cdot(k+1)}{2}} \tag{3.9}$$

3.3 Hypotheses Based on the Bias Formula

From the derived formula for the skew influenced by the bias we hypothesize the following relations. More bias means that the skew between the reference node and a chosen node in the line is closer to zero.

If the number of hops is increased the bias at the last node should increase as well since $\alpha < 1$ and k is increased. The bias should also increase if the variance of the noise increases because this would decrease α.

If the synchronization period is increased the variance of the regression table entries increases as well (c.f. relation between variance and synchronization period in Section 4.2.2). This in turn decreases the influence of the variance of the noise in α. This means that the bias α increases and this should lead to a skew that is less biased (i.e. the influence of the bias is smaller). The same effect should be observable if the number of regression table entries is increased since this increases the variance of the regression table entries, too. All the hypotheses are summarized in Table 3.1.

Table 3.1: Summary of the Hypotheses

	Skew ($\beta^{Ref\rightarrow k}$)	**Influence of the bias**
Number of hops $k \uparrow$	$\downarrow$	$\uparrow$
Noise variance $\sigma_\eta^2 \uparrow$	$\downarrow$	$\uparrow$
Synchronization period $\Delta x \uparrow$	$\uparrow$	$\downarrow$
Number of entries in the regression table $n \uparrow$	$\uparrow$	$\downarrow$

Chapter 4

Tests on the Testbed

In order to examine whether the formulated hypotheses describing the expected behavior of FTSP running on a real WSN we conduct tests on a testbed. To analyse the behavior offline we collect traces of the running FTSP protocol.

We use the the obtained traces to analyse the values of the metrics described in Section 3.1. In addition we use the data to estimate the parameters for the simulation described in Chapter 5 and the formula described in Section 3.2.

First we describe how the traces are collected using the testbed. Then we present and discuss the results obtained from the collected traces.

4.1 Collecting Traces from the FlockLab Testbed

This section describes the setup we used to collect the traces from running FTSP on the testbed.

First we list the tools used for the collection. Then we explain the topology used. Finally we describe the implementation which logs the data.

4.1.1 Tools

In order to obtain traces for further analysis we use the FlockLab testbed presented by Lim et al. in [6] installed at ETH Zurich. Every sensor node in the network of this testbed has its own observer node. These observers allow to easily deploy executables to all nodes in the network and to obtain logs of all serial messages logged by all the nodes. Furthermore the observers allow to set a GPIO pin at all sensor nodes simultaneously.

For the experiment on the testbed we use the Tmote Sky sensor node [2]. It features a 8 MHz microcontroller, a 32 kHz timer and a 2.4 GHz radio transceiver. To program the Tmote Sky sensor nodes we use the TinyOS operating system [1]. TinyOS is an open source operating system which is event driven and modular. It is widely used with WSNs. In addition we use the FTSP implementation provided by TinyOS.

In order to process the raw data obtained from the FlockLab testbed and to analyse it we use Python.

4.1.2 Topology

We chose the simplest topology which allows to analyse the multi-hop behavior: a line topology as depicted in Figure 4.1. The first node in the line is the reference node. For this node it holds that the local time is always equal to the global time[1]. To ensure this topology in a reliable way we slightly modified the FTSP protocol such that a particular node only accepts time synchronization messages from one specific node, its preceding node in the line. This makes it possible to place more than two nodes in each others radio range and they still only process messages from the defined preceding node. The testbed network of nodes including the topology used in the experiments is depicted in Figure 4.2. The network consists of 27 nodes whereof the first node is the reference node.

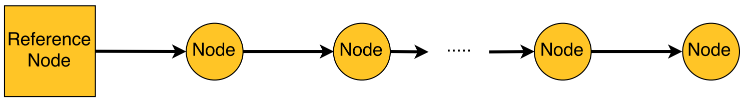


Figure 4.1: Line topology

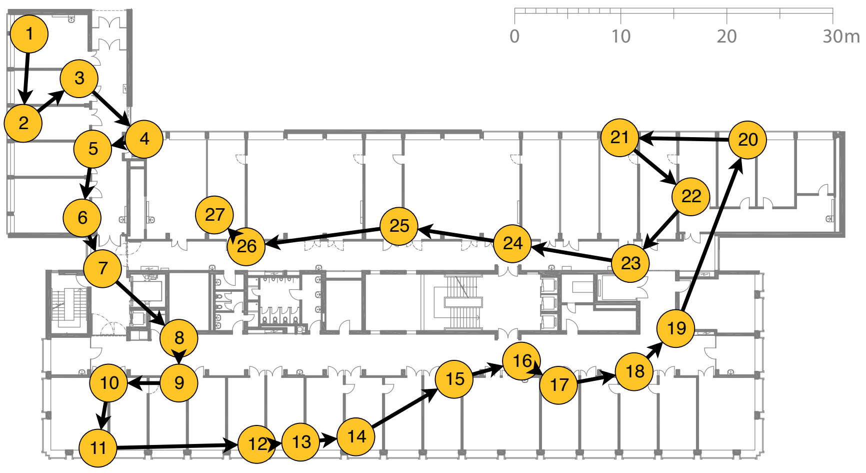


Figure 4.2: Line topology implemented with the sensor nodes of the FlockLab testbed

4.1.3 Implementation of the Logging Application

In order to collect data from FTSP running on sensor nodes we modified the FTSP implementation such that it allows to collect many values used and calculated by the FTSP protocol. In addition we created an application that uses the modified FTSP implementation and logs the collected data. With the application running on the FlockLab testbed we log three different types of data sets: Synchronization Points, Sent Timestamps and GPIO Events.

Synchronization Points

Every time a synchronization point from an arrived synchronization message is added to the regression table the node logs all values from corresponding synchronization message and all updated state values of FTSP after having incorporated the synchronization point. The synchronization message values include the latest synchronization point, the ID of the sending node as well as the sequence number of the time update. The state values include all synchronization points currently in the regression table as well as the parameters (skew and offset) from the regression calculated by the sensor nodes.

[1]Note that this is not true in general with FTSP. If a node that previously calculated the parameters (skew and offset) from the regression table entries becomes the root it will continue to use these parameters to calculate the global time. However in the experiments the first node (root node) is never synchronized to any other node because there is no node with a lower sequence number.

Sent Timestamps

The synchronization messages do only include the global time but not the local time of the sending node. Therefore the nodes log the local timestamp every time they send a synchronization message. This data is then used to figure out the local timestamp corresponding to the global timestamp in the synchronization messages. The matching of the timestamps is done offline.

GPIO Events

We configured the FlockLab testbed to periodically produce a GPIO event for all sensor nodes simultaneously. This GPIO event is then used to log the local time of all nodes at the same time instance. This data allows the investigation of the clock deviations of the different nodes. In addition it allows to compare the estimated global time of all nodes.

Precision of the Timestamps

The timestamps in our implementation have a resolution of 32.768 kHz. The timestamps of the GPIO Events are produced with an interrupt but do not make use of an input capture because the sensor platform node used in this thesis doesn't support this. The accuracy of the GPIO timestamps is in the order of 1 microsecond [7].

Implementation of Alternative Linear Regression Algorithms

Most of the WSN platforms do not support high precision floating point variables and operations. The reason is that the floating point operations are expensive in terms of clock cycles since most such devices do not have a dedicated floating point unit in hardware. This makes it quite hard to implement more complicated mathematical algorithms in an efficient manner and yet to provide enough accuracy. Even the implementation of the simple OLS regression of FTSP uses integer division wherever possible.

4.2 Results

In this section we present the results obtained from the traces from the FlockLab testbed when running FTSP.

4.2.1 FTSP Performance

Plots of the time difference to the global (reference) time and the skew are depicted in Figure 4.3 and Figure 4.4. The plots both show a boxplot with 516 samples taken at a time interval of 5 seconds. As expected the variance of the time difference / the skew increases with the number of hops because of the error accumulation [5]. The error accumulates mainly because the estimate of the global time deviates from the actual global time (even though linear regression is used) until the node receives a new synchronization point [5].

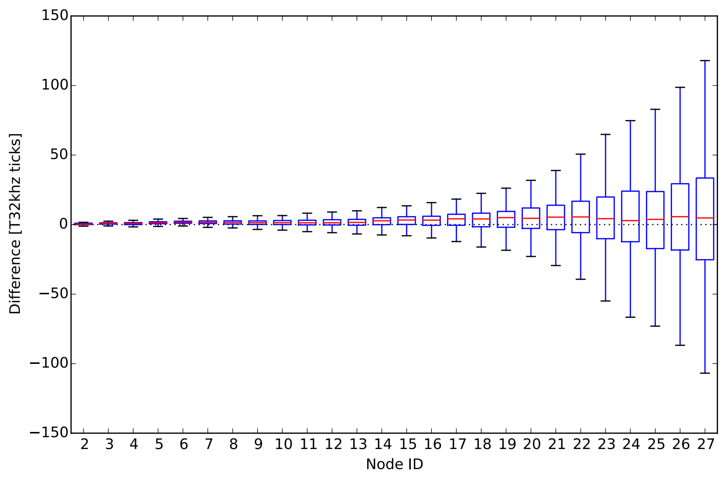


Figure 4.3: Time difference to the global time of all nodes

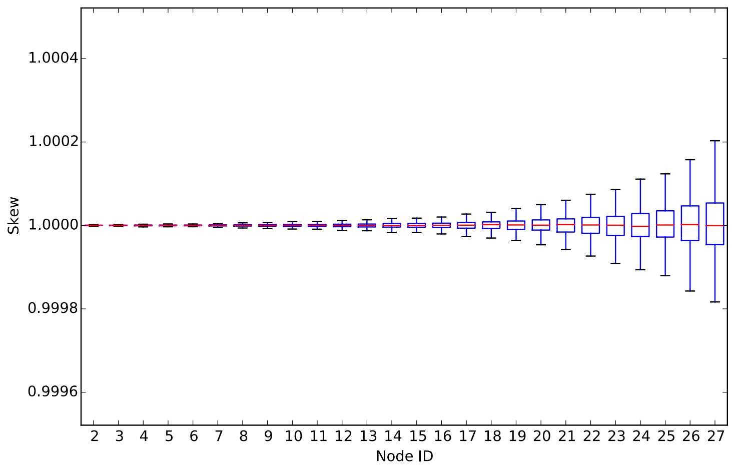


Figure 4.4: Skew of all nodes

Convergence Time

In order to analyse the results of the tests on the testbed it is important to know when all nodes are synchronized to the reference node. In the optimal case the convergence time can be calculated from the parameters of the FTSP protocol by using the formula stated in [8] and described in Section 2.3.3:

$$t_{cs} = P \cdot (M + N \cdot R) \tag{4.1}$$

In the experiments the following parameters have been used:

$P:$	3.0 s
$M:$	5 sync periods
$N:$	3 entries
$R:$	26 hops

This results in a convergence time of $t_{cs} = 249\,\mathrm{s} \approx 11\,\mathrm{min}$ in the ideal case. This assumes that no message is corrupted and no node clears its regression table.

In Figure 4.5 the time until the node knows for the first time the ID of the reference node and the time until the synchronization is steady is depicted for all nodes. The results of the testbed show that the convergence time is about 900 s with the line topology. This means that our setup and / or the line topology is not ideal. In the transient phase the nodes change their known ID of the reference node frequently. Reasons for this transient phase are that packets are lost (see section 4.2.1) or the previous node is not yet in a steady state and therefore sends misleading synchronization messages.

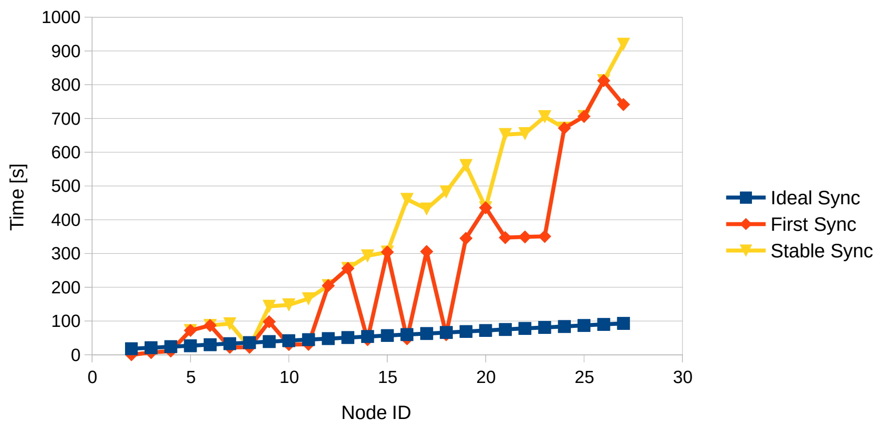


Figure 4.5: Time until sensor nodes are synchronized to the reference node

It may seem strange in Figure 4.5 that a node further to the end of the line knows the root ID before a node further to the reference node knows the root ID. This can be explained by the fact that in that figure only data from the synchronization update messages that are added to the regression table are plotted. The node however updates the information about the current root ID with every synchronization message that arrives (including messages that are not used in the regression table). This means the information about the root ID can become true and again false before the first synchronization message with the correct root ID is added to the regression table.

The convergence time in real WSNs might be better since normally no line topology but a mesh topology is used. This means that there are redundant transmissions of synchronization messages with the same sequence number. Therefore multiple synchronization messages should reach a certain sensor node and less synchronization points should be lost.

In the results we show in this chapter we remove the first 16 minutes to skip the phase when the nodes are not synchronized to the reference node.

Decrease of Fresh Synchronization Messages

From the results of the tests on the FlockLab testbed we observed that even if the network converged there are intervals between the timestamps when a synchronization point has been added to the regression table which are larger than the synchronization period.

In Figure 4.6 we depicted various counts of sent and received messages. *Messages ideally sent/received* corresponds to the maximum possible number of messages that could have sent within the given time interval and with the given synchronization period. Note that the FTSP protocol doesn't use the synchronization period parameter directly but modifies this value by up to $\pm 15\,\%$. Obviously the pseudo random number generates exhibits a strongly repetitive pattern. Most likely this randomness should prevent too many packet collisions at the receivers. The line labeled *Messages sent* corresponds to the actually sent messages. This value is derived from the Sent Timestamps data set. Because this line matches the first line very closely it can be concluded that all nodes send synchronization messages very regularly without skipping many messages.

However this line doesn't tell us how many of these sent messages are "fresh". With "fresh" we mean messages which carry a sequence number that has not been sent by the same node

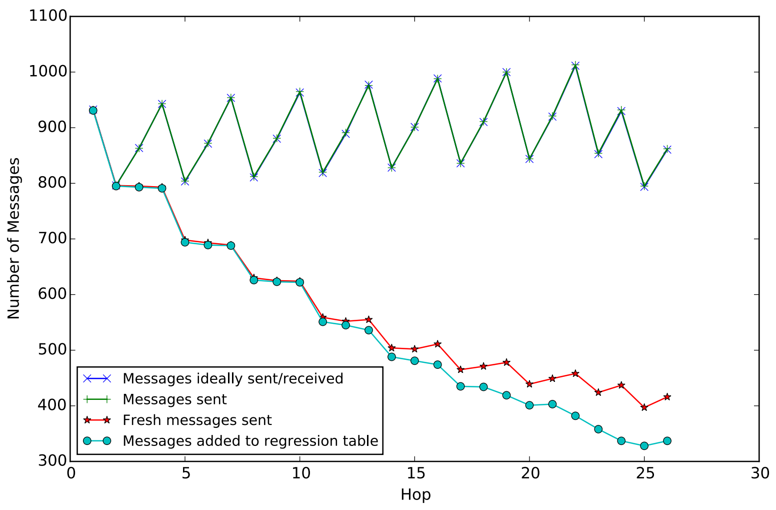


Figure 4.6: Number of sent and received messages

before (excluding the overflow of the sequence number field). The number of fresh messages is represented with the line *Fresh messages sent*. This line shows that quite a lot of sequence numbers in the messages are repetitions of previously sent messages. A node does not increase the sequence number in the message if it didn't add a new message with a higher sequence number to the regression table. This can happen if the node didn't receive a message with a higher sequence number or if it didn't receive any message in that synchronization period. As depicted in Figure 4.7 the difference in synchronization periods between the nodes causes situations where a node with a smaller synchronization period (node b) sends 2 messages before its predecessor node with a larger synchronization period (node a) sends a message with a new sequence number. Because the difference in synchronization periods is quite significant this could happen quite often and could lead to a lot of messages with non-increased sequence numbers. In the case of a line topology (as it is the case in our experiments) a repetition of a sequence number is propagated through the rest of the line. In a real network this problem is most likely less severe since one node may receive messages from multiple stations and only one reception of a message with a higher sequence number is enough to remove the repetition.

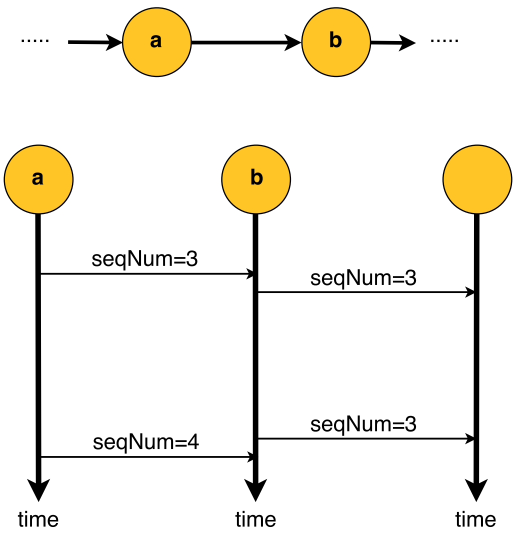


Figure 4.7: Different synchronization periods leads to repetitions of sequence numbers

The last line in Figure 4.6 labeled *Messages added to regression table* corresponds to the number of messages that have been received and have been added to the regression table. If no message would have been lost this number should equal the number of fresh messages sent. So the difference is the number of actually lost messages. The ratio of lost message over fresh messages sent is depicted in Figure 4.8. A possible reason for messages being lost is a bad channel which disrupts the transmission of the packets. A corrupted packet is detected by a CRC checksum that doesn't match the packet. However the problem of corrupted or lost messages could be much less severe in a real use case since normally there is no line topology but a mesh topology. In that case synchronization messages with the same sequence number are usually received many times which provides redundancy.

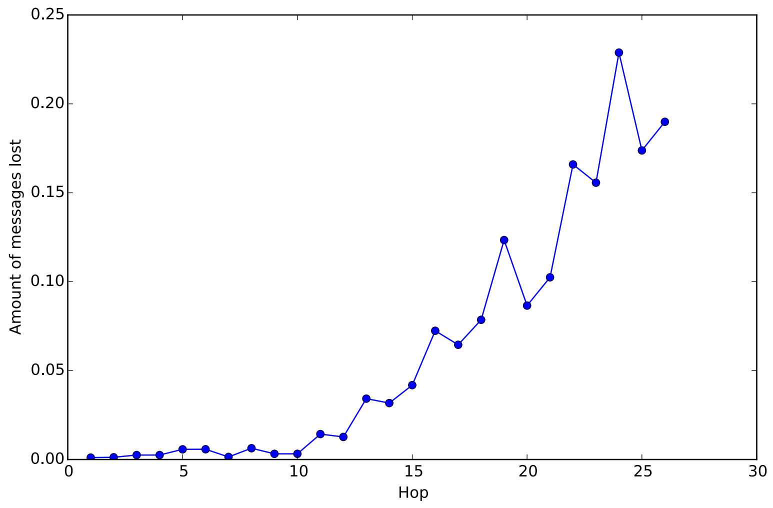


Figure 4.8: Amount of lost messages

The messages with not-increased sequence numbers and corrupted messages are a problem in order to investigate the accuracy of alternative linear regression algorithms. As showed in Section 3.2 the bias of OLS regression depends on the variance of the regression table entries. However if this variance is not stable and cannot be controlled it is rather difficult to make any statements of how good a certain linear regression algorithm is compared to the OLS implementation of FTSP. It might be possible that the increasing variance of the regression table compensates the bias of OLS to some extent. But this is a speculation only.

Oscillations

Another phenomenon we observed by analysing the FlockLab test data is the oscillation of measured time difference / skew over time at a node. An example for such a behavior is depicted in Figure 4.9 and Figure 4.10. Tests and simulations with different parameters showed that the period of these oscillation depend on the size of the regression table used by the FTSP protocol. Figure 4.11 which corresponds to a simulation with a regression table size of 8 entries shows small oscillation periods. Whereas the Figure 4.12 which corresponds to a regression table size of 32 entries include much longer oscillation periods.

This can be explained by the following scenario: A newly added synchronization point differs significantly from the synchronization points already in the table and the synchronization points of the near future but is still in the tolerance range to accept it as update in the table. The deviation of this point influences not only the current regression but all regressions until it will be evicted out of the table. In addition parameters calculated from the regression also differ from the past and the future parameters. This means one exceptional synchronization point affects multiple parameters over time and also the following nodes. Several such exceptions occur during the propagation of the global time through the line of nodes. The observed oscillations then are the result of the superposition of all such events. This explains why this effect is much stronger towards the end of the line.

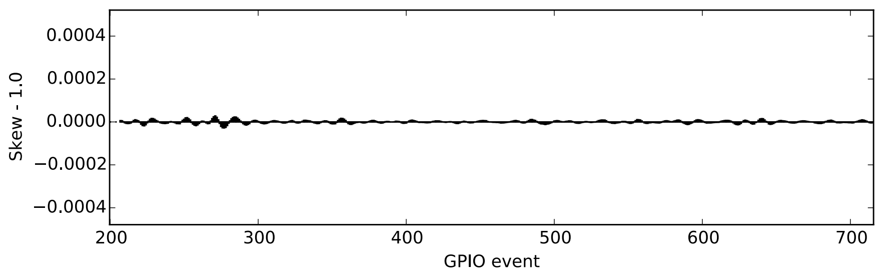


Figure 4.9: Small oscillations at node 14 (testbed data)

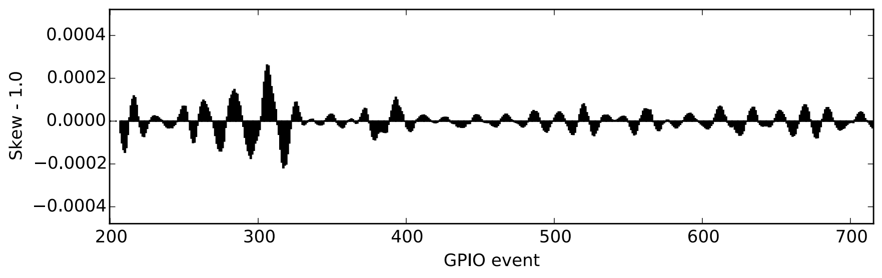


Figure 4.10: Large oscillations at node 24 (testbed data)

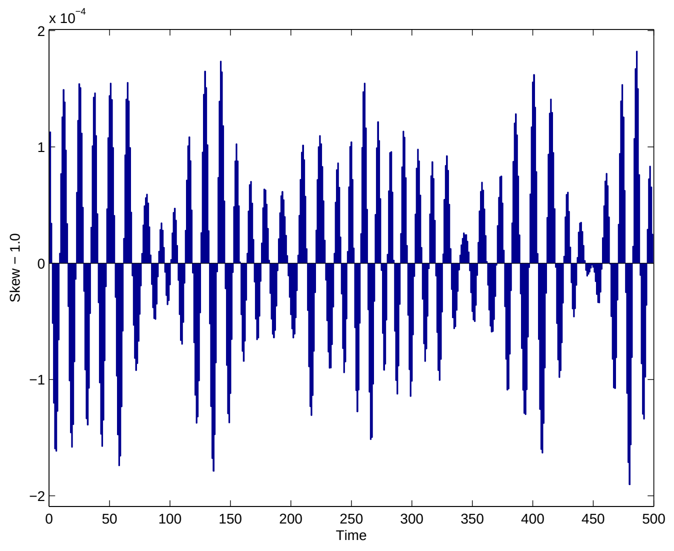


Figure 4.11: Short oscillations periods with 8 entries in regression table (simulation of node 8)

The RMSE of the regression of the regression table entries is depicted in Figure 4.13. The results show that the peaks of the deviations correlate quite well with the peaks of the RMSE. Since the RMSE corresponds to the residuals this shows that the oscillations depend on the size of the residuals.

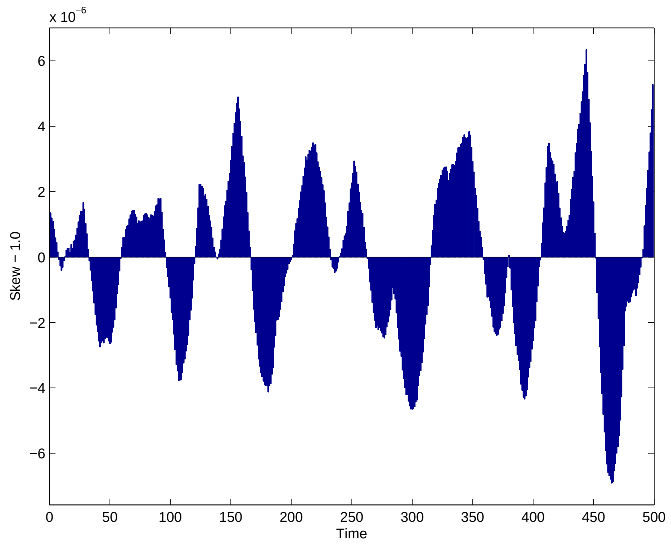


Figure 4.12: Long oscillations periods with 32 entries in regression table (simulation of node 8)

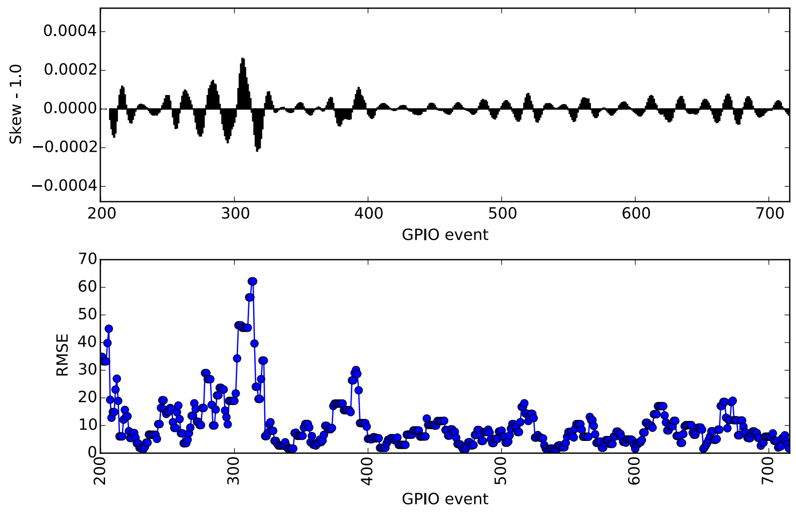


Figure 4.13: Large deviations of the skew correlate with large RMSE (testbed data)

4.2.2 Parameters for Simulation / Bias Formula

We used the data from the tests on the testbed to obtain estimates of important values for the simulation and the formula to calculate the bias. The estimated values are the variance of the noise of the clocks, the variance of the drift of the clocks and the variance of the regression table entries (local timestamps).

To estimate the variance of the noise and the drift produced by the clocks on the sensor nodes we used the timestamps obtained from the GPIO events and performed an OLS regression with the timestamps from the examined node and the timestamps from the reference node. The model then assumes that the residuals of this regression correspond to noise measurements and the slope corresponds to drift measurements.

An exemplary plot of the noise estimation for node 8 is depicted in Figure 4.14. The mean is at 0.0 as expected. The variance is very low at around 1.5 ticks[2].

The result of the drift estimation is depicted in Figure 4.15 and Figure 4.16. As expected the

[2]A *tick* corresponds to one clock cycles. In this case one tick corresponds to $1/32'768$ seconds

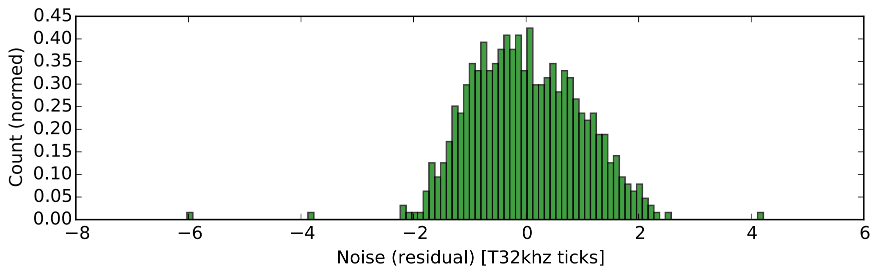


Figure 4.14: Estimated noise of the clock of node 8

mean is around 1.0. The variance is very low with a value of 1.13e-09. The result shows that most of the nodes are very similar. However there are three nodes which stick out. Node 17, 19 and 21 exhibit a significantly higher drift. This behavior is consistent with observations from other experiments that run on the same testbed. It seems that the design or the manufacturing of the nodes is different. Most likely they use a different type of quartz crystals as time source.

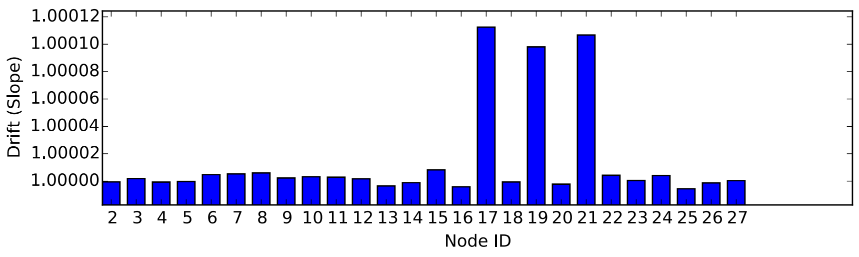


Figure 4.15: Estimated drift of all nodes

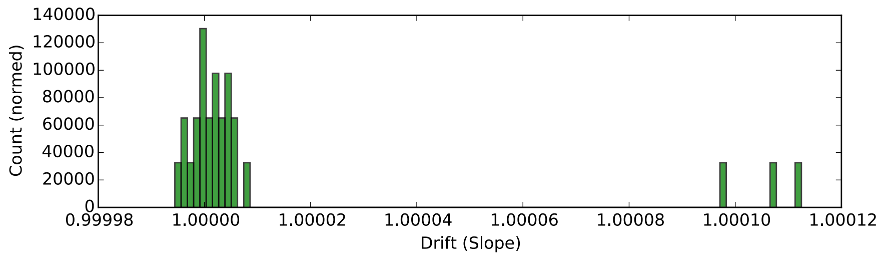


Figure 4.16: Histogram of the estimated drift

Another important value for the simulation and the use in the formula is the variance of the regression table entries (more precisely the local timestamps). From the design of the FTSP protocol one would expect this value to be constant for all nodes since the sync period (rate with which the synchronization points are sent) and the regression table size is the same for all nodes. However the data from the tests on the testbed shows that this assumption is wrong for a real FTSP implementation. Figure 4.17 shows that the mean variance of the regression table entries increases almost monotonically with the number of hops.

This behavior can be explained by the fact that there are messages with a non-increasing sequence number and by the fact that messages get disrupted which is described in Section 4.2.1. The relation between the variance of the regression table entries (σ^2), the synchronization period (Δx) and the number of entries in the regression table (n) is given by the following formula:

$$\sigma^2 = \Delta x^2 \cdot \frac{n^2 + n}{12} \tag{4.2}$$

The variance of the regression table entries increases with the number of entries in the regression table and with the synchronization period. For the tests used to estimate the parameters

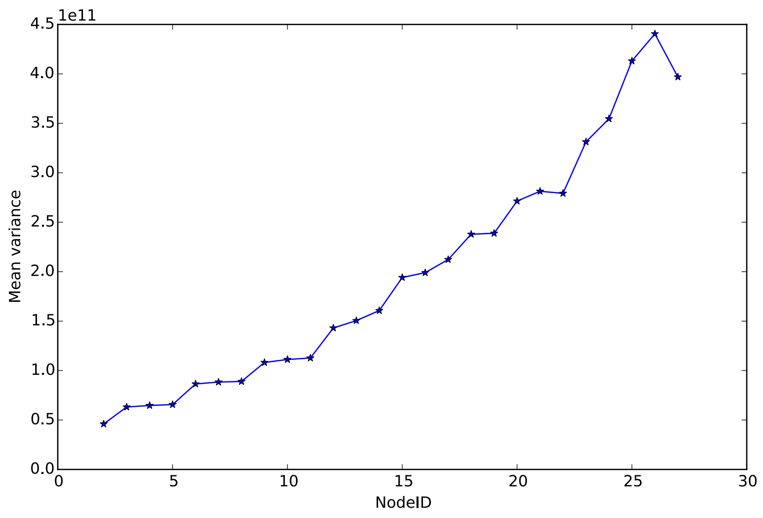


Figure 4.17: Mean variance of the regression table entries for all nodes

for the simulation the number of entries in the regression table is $n = 8$ and the synchronization period is set to $\Delta x = 3.0\,\mathrm{s} \cdot 32'768\mathrm{ticks/s} = 98304\,\mathrm{ticks}$.

For the ideal case (e.g. first hop) the variance would be $\sigma^2 = 5.7 \cdot 10^{10}$. This value matches quite well the value in Figure 4.17 for node 2. Figure 4.6 shows that in only about every 2.4-th (40 %) of all possible sync periods a synchronization point is added to the regression table on average. This increases the effective synchronization period to about $3.0 * 2.4 = 7.2$ on average at the last node in the line. This would correspond to a variance of $3.3 \cdot 10^{11}$. Again this value matches the value at the last node in Figure 4.17 quite well.

Chapter 5

Simulation

Because we could not observe all relations derived in the formal analysis in Section 3.3 on the testbed we decided to build a simulation that simulates the FTSP protocol. The advantage of the simulation over the real testbed is that it has no hardware limits. The simulation allows to simulate a network with an arbitrary number of hops whereas with the testbed only 27 hops can be used at maximum.

5.1 Parameters of the Simulation

The relevant parameters for the simulation are the protocol parameters as well as the variances of the clock noise and the clock drift of a real WSN. The protocol parameters in general are the same as used in the tests on the testbed. The variances of the real WSN have been estimated in Section 4.2.2 based on the data obtained from the testbed. Furthermore there are simulation specific parameters. All the parameters are listed in Table 5.1, Table 5.2 and Table 5.3.

Table 5.1: Protocol parameters

Synchronization period Δx	$3.0\,\mathrm{s} = 98304\,\mathrm{ticks}$
Table size n	$8\,\mathrm{entries}$
Number of hops	$30\,\mathrm{hops}$

Table 5.2: Values estimated by data obtained from the testbed

Variance of clock noise	1.5
Variance of clock drift	$1.13 \cdot 10^{-9}$

Table 5.3: Simulation specific values

Number of updates	1000 (not including discarded updates)
Number of discarded updates	100
Initial local / global time	$30'000'000\,\mathrm{ticks}$

The variance of the regression table entries is determined by the sync period and the table size (c.f. Equation 4.2).

5.2 Design of the Simulation

To create the simulation we use MATLAB. The simulation mainly simulates the time synchronization from hop to hop. For this it uses a line topology as the tests on the testbed do. It simulates the noise and the drift of the nodes by using a normal distributed random values. As in the real WSN the time synchronization messages from different synchronization rounds propagate through the network concurrently. In order to simulate the linear regression algorithm of FTSP as good as possible the simulation uses a C program that implements the line-fitting algorithm of FTSP. However to compare OLS and GMR we use linear regression algorithms from MATLAB.

The simulation differs in some points from the real tests on the testbed. First the nodes are initialized with optimal values (i.e. parameters that map the local time to the exact global time). In a real testbed this cannot be assumed. To mitigate the effect of this difference we discard the first 100 updates produced by the simulation. Another difference is the synchronization period. In the simulation the exact same value is used for all nodes. In addition the points in time when the synchronization takes place in the simulation are exactly the same for all nodes. In a real WSN the FTSP randomizes the actually used synchronization periods slightly (by about 15 %) and hence the points in time when the synchronization takes place is not the same for all nodes. A further difference is that the variance of the regression table entries is constant over all hops. As the results in Section 4.2.1 show this doesn't apply for a real WSN where synchronization messages with not-increased sequence numbers are produced.

The obtained results are analysed using the same metrics as used to anlyse the data from the real testbed. The most important value is the the number of hops which are necessary such that the bias of OLS deviates the skew such that it equals 0.5.

5.3 Results

In this section we present the results obtained by the simulations. First we show the difference of OLS and GMR using the MATLAB regression algorithms. Afterwards we present simulations using the estimated parameters. For this we use the parameters stated in Section 5.1 as initial settings. Then we modify the parameters one by one to observe the effect corresponding to the parameter. Finally we present plots of the Bias formula using the estimated parameters in order to compare the formula with the simulation.

5.3.1 Comparison of OLS and GMR

The simulation allows to easily compare the two linear regression algorithms of interest. Figure 5.1 (with artificial noise) clearly shows that OLS regression suffers from a bias towards zero. With GMR the bias is significantly smaller. This shows that the simulation model can be used to investigate the bias property of the linear regression algorithms.

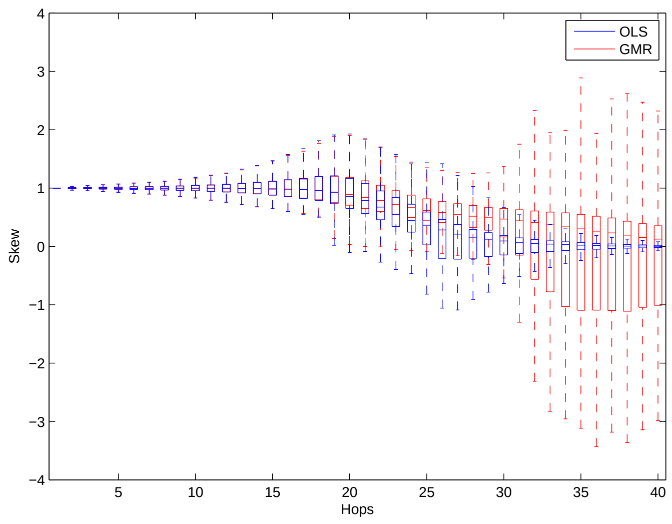


Figure 5.1: Simulation of OLS and GMR

5.3.2 Using the Estimated Parameters

The results from the simulations with the estimated values of the variances and the same parameters as used in the tests on the testbed show that there is no bias in the first 30 hops (c.f. Figure 5.2).

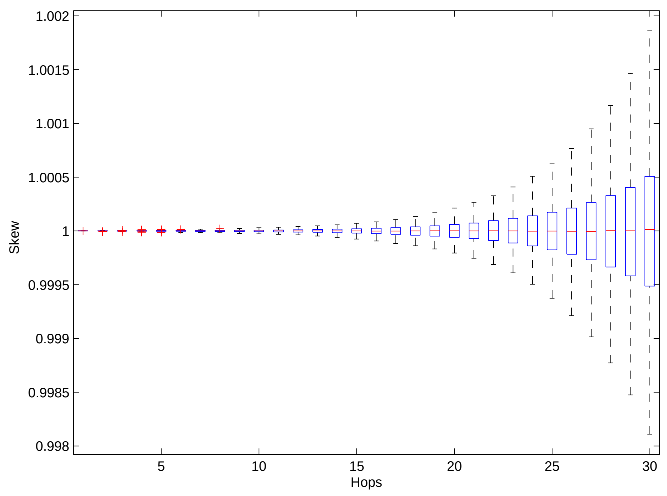


Figure 5.2: Simulation with 30 hops using the estimated values as input

However if the number of hops is increased the bias becomes visible after approximately 65 hops (c.f. Figure 5.3). This means in a WSN with more than 65 hops the use of GMR would be beneficial.

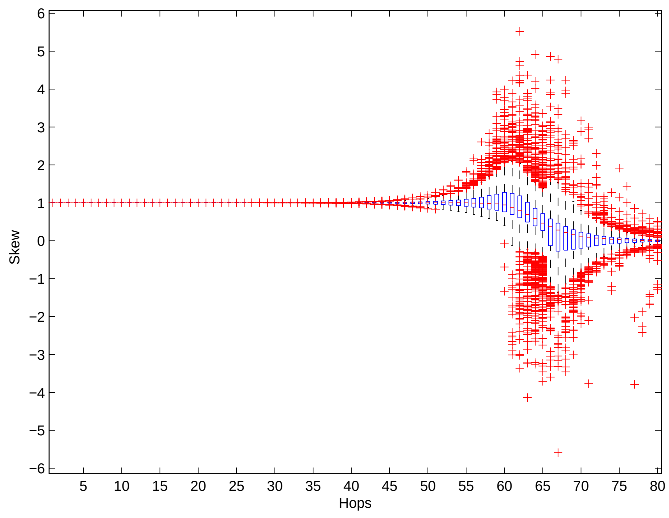


Figure 5.3: Simulation with 80 hops using the estimated values as input

According to the theory another way of increasing the bias is to increase the noise. The results of a simulation with artificially increased noise (the other parameters are unchanged) is depicted in Figure 5.4. The bias of the OLS regression is clearly visible within the first 30 hops.

This means that that sensor nodes of the testbed exhibit not enough noise and the tests include not enough hops such that the bias of OLS regression matters. Therefore in the tests on the testbed it is not beneficial to use GMR instead of OLS. However if the noise and/or the number of hops is larger it would be advantageous.

Figure 5.5 shows simulation results when using the estimated parameters but setting the synchronization period to 0.003 s. With these parameters the bias is again observable in the first 30 hops. This shows that a smaller sync period leads to a larger influence of the bias. How-

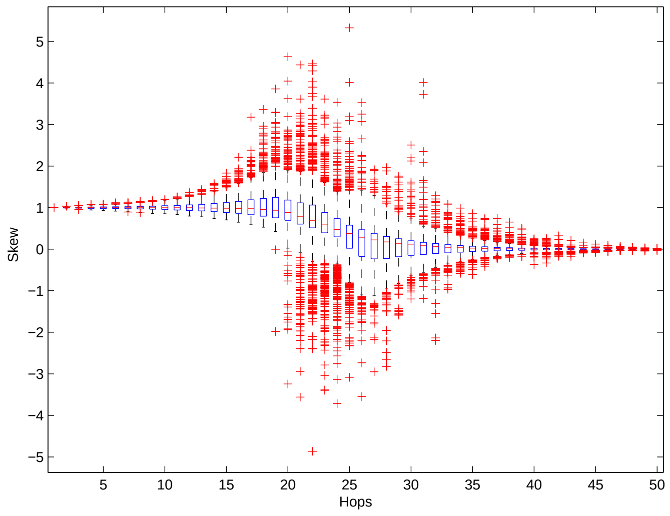


Figure 5.4: Simulation with 50 hops artificially increasing the noise (other parameters as in Figure 5.3)

ever the sync period is about 1000 times smaller compared to the default setting depicted in Figure 5.3.

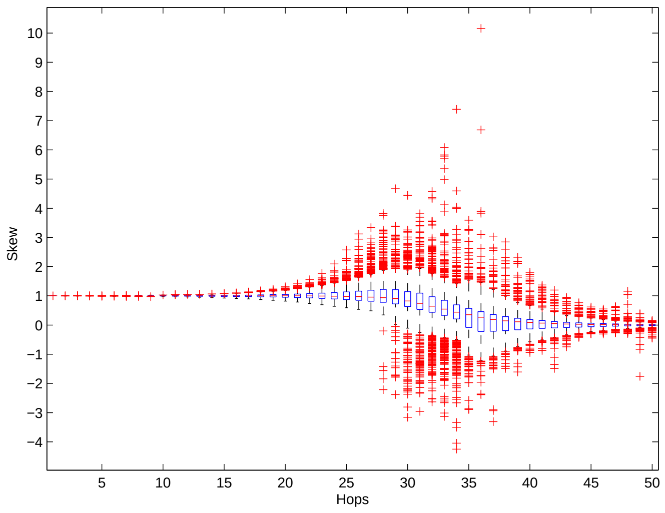


Figure 5.5: Simulation with 50 hops with a synchronization period of 0.003 s (other parameters as in Figure 5.3)

Figure 5.6 shows the result of a simulation using 4 instead of 8 entries in the regression table. The other parameters of the simulation are the same as in Figure 5.3. The simulation implicates that the skew equals 0.5 not before the 90-th hop. The comparison of Figure 5.6 and Figure 5.3 shows that increasing the number of entries in the regression table leads to a larger influence of the bias. This contradicts the relations derived from the mathematical formula in Section 3.1.

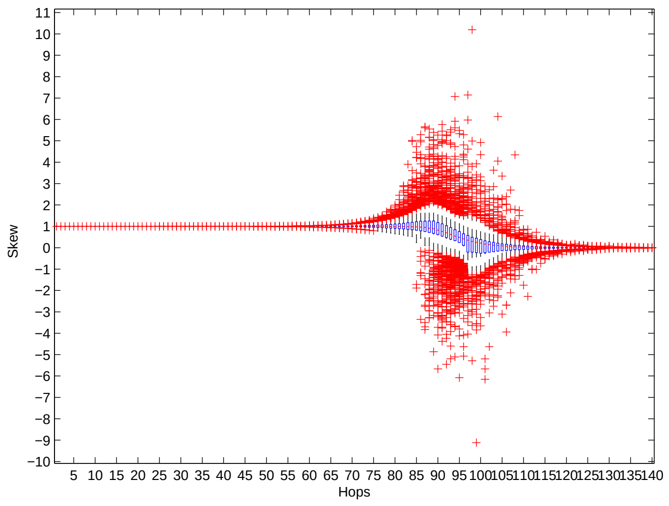


Figure 5.6: Simulation with 80 hops with 4 entries in the regression table (other parameters as in Figure 5.3)

Bias Formula

In Figure 5.7 the result of bias formula when using the estimated parameters obtained in Section 4.2.2 is depicted. Unfortunately the formula doesn't match the simulation well. The result states that the the skew equals 0.5 if a line topology with 230'000 hops is used.

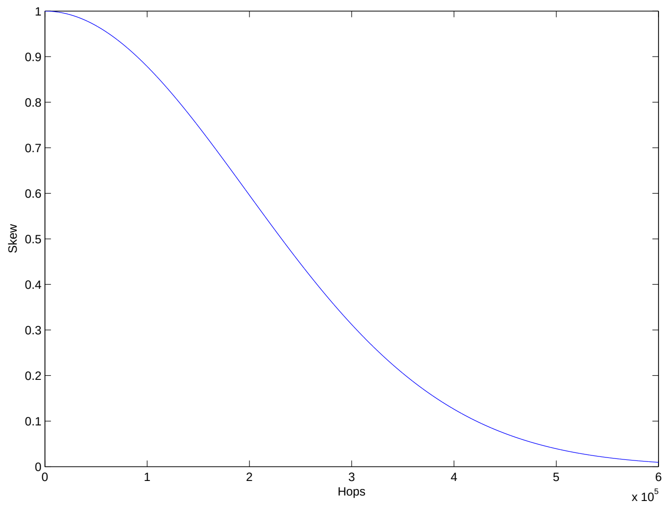


Figure 5.7: Skew calculated with the bias formula with estimated parameters

If the noise is increased the formula matches the simulation better. This is depicted in Figure 5.8. The result states that the the skew equals 0.5 if a line topology with 28 hops is used.

The discrete timestamp values (integers) of the regression table entries influence the result of the formula significantly as a simple simulation with rounded values in Figure 5.9 shows. This can explain part of the inconsistency between the simulation and the bias formula.

Another part could be explained by the assumptions made in the derivation of the bias formula. With the assumption that all $\beta_*^{a \to b} = 1$ the drift of the clocks is neglected. However this might not have much influence. Another assumption is that the variance of the table is constant which according to the results from the testbed is not the case. However in the simulation this variance is constant. Therefore this assumption does not explain the inconsistency between the simulation and the bias formula.

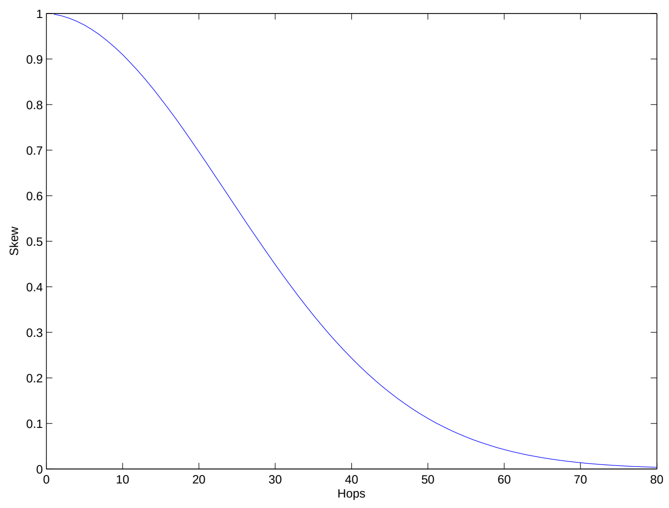


Figure 5.8: Skew calculated with the bias formula with large noise variance

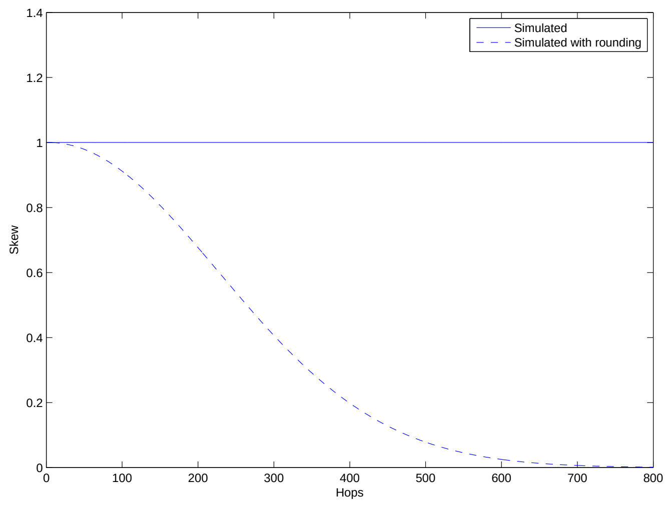


Figure 5.9: Skew calculated with the bias formula with and without rounding

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this semester thesis we successfully obtained traces from FTSP running on a testbed. We analysed the performance of the time synchronization protocol with special attention on the linear regression algorithm. Furthermore we simulated the behavior of OLS and an alternative linear regression algorithm, GMR.

The analysis showed that the nodes cannot update their estimate of the global time as often as the protocol would suggest. When using a line topology a large amount of messages provides no fresh synchronization points and another significant amount of messages is lost because of bad transmissions. This behavior influences the quality of the linear regression used to calculate the parameters to map the local time of a node to the global time of the network.

The use of an alternative linear regression algorithm with properties to reduce the error accumulation is a promising idea. However it turns out that the current implementation of the linear regression in FTSP is very good for networks with sensor nodes exhibiting only low noise and networks with less than 50 hops. Simulations showed that only nodes with clocks which exhibit a lot of noise or networks with many hops (more than about 50 hops) would benefit from using GMR. In addition we learned that it is challenging to implement GMR efficiently and with high precision on resource constrained sensor nodes. The limitations of sensor nodes often do not allow to use high precision floating point variables and operations.

6.2 Future Work

Formal Analysis

The bias formula provides relations of which many can be confirmed by the simulation. However it cannot accurately estimate the skew and seems to be sensitive to rounding. In general it seems that it is rather difficult to model all the inaccuracies of the real system with a simple formula. Nevertheless maybe a more advanced formula / model could be found to estimate the bias for a real system.

It would be interesting to investigate why the hypothesis which states that increasing the number of regression table entries leads to a smaller bias could not be confirmed. Its possible that the hypothesis is wrong or maybe the simulation doesn't allow to correctly simulate this relation.

Tests on the Testbed

The implementation of the logging program which runs on the sensor nodes could be improved by using a more accurate method of time-stamping the GPIO events. It could for example use a capture register which enables to get a precise timestamp when an GPIO interrupt occurs. Unfortunately the Tmote sky sensor node used in this thesis doesn't offer this method.

An idea which allows to implement linear regression algorithms in a standard way is the use of a multiple precision library. This would allow to use floating point numbers and floating point operations and having as much precision as desired.

Another interesting extension would be the implementation of a loop topology in order to emulate more than 27 hops with the given testbed. This would hopefully allow to see the bias on the testbed. This approach is however not unproblematic since this would very likely implicate more message collisions.

Yet another idea is to conduct tests with a topology which includes some redundancy. It would be interesting to see whether this actually reduces the amount of messages with not-increased sequence numbers.

Another promising idea is to enforce the same synchronization period for all nodes. One could even try to increase the synchronization period monotonically with the hop distance to the reference node. This should reduce the amount of messages repeat a previous sequence number, too. With this the variance of the regression table entries should increase less towards the end of the line.

Ideally one could remove the increasing variance of the regression table entries completely in order to eliminate all potential effects of this behavior. Together with a network allowing for more hops one could either confirm or reject the speculation that the increasing variance compensates the bias.

Simulation

The simulation could be extended in order to simulate the time synchronization protocol more accurately. For example different synchronization periods for different nodes could be useful. This would allow to simulate an increasing variance of the regression table entries by decreasing the synchronization period with the hop-distance from the reference node. In addition the simulation could include more non-ideal behavior to be even more realistic. For example one could simulate a message loss rate.

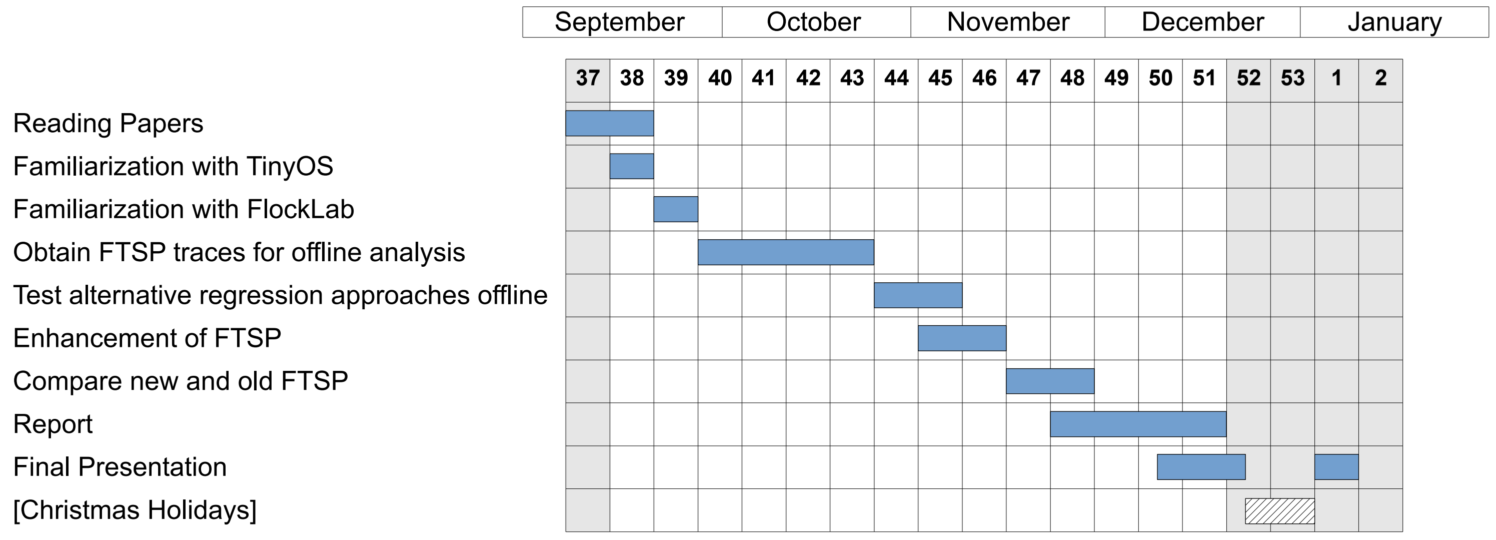
Appendix A

Time Schedule

See next page.

project_plan_v1.1, 02.11.15 Semester Thesis Roman Trüb

Project Plan



▨ Christmas Holidays: 24.12.15 – 2.1.16

Appendix B

Project Description

See next pages.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autumn Semester 2015

Semester Thesis

for
Roman Trüb

Supervisors: Roman Lim
Olga Saukh

Start date: September 14, 2015
Hand-in date: December 18, 2015

Low-Error Multi-Hop Clock Synchronization

Introduction

Maintaining precise time synchronization is essential in wireless sensor networks. However, clock sources often exhibit severe drift. Also message exchange is subject to delays due to varying distances between nodes, packet collisions, and multipath effects. Sophisticated time synchronization algorithms are necessary to keep nodes' clocks synchronized. Existing network-wide synchronization algorithms for sensor networks synchronize clocks hop-by-hop. This leads to a well-known problem of error accumulation over multiple hops and results in large synchronization errors at nodes with increasing distance from a reference time source.

State-of-the-art time synchronization protocols for wireless sensor networks (e.g., Flooding Time Synchronization Protocol (FTSP[1]) and Rapid Time Synchronisation (RATS) [2]) use ordinary least squares regression to estimate and compensate clock drifts. We have good arguments to believe that this standard regression approach is not appropriate in the context of time synchronization and suggest using a different regression approach. In a setting related to sensor calibration [3] it has been shown that geometric mean regression allows to reduce multi-hop calibration errors, leading to a significantly better calibration of all sensors in the network.

Objective

The goal of this thesis is to enhance FTSP – a popular time synchronization protocol for sensor networks – with an alternative linear regression implementation and compare its performance to the original implementation. You will first test alternative linear regression methods on a set of real traces gathered with FlockLab [4], our local testbed. Then, you will modify the TinyOS FTSP implementation running on Tmote Sky sensor nodes to integrate a new linear regression approach and optimize your code for efficiency and high precision computation.

Task Description

1. Create a project plan and determine milestones, both time wise and topic wise. Care should be taken to allow for ample time for the final presentation and the report.
2. Make yourself familiar with the relevant work related to multi-hop time synchronization in wireless embedded systems.
3. Download and install TinyOS and get familiar with this operating system that can be used with TelosB nodes.
4. Run FTSP on the FlockLab testbed to obtain a set of communication traces for further off-line analysis.
5. Study alternative regression approaches and apply them to the traces acquired in the previous step, e. g., using MATLAB.
6. Enhance FTSP with an alternative regression approach to compensate for clock drift and offset. Optimize your implementation for efficiency.
7. Compare your implementation to the original protocol implementation.
8. Document the work done thoroughly by means of a presentation and a written report.

Thesis Organization

- Duration of the work:
  The semester thesis starts on September 14, 2015and ends on December 18, 2015.
- Project plan:
  A project plan with milestones is held and updated continuously. Unforeseen difficulties that change the project plan are documented and discussed with the supervisors in a timely manner.
- Weekly meetings:
  In regular (weekly) meetings with the supervisor, the current state of the work, potential difficulties, as well as future directions are discussed.
- Initial presentation:
  Two to three weeks after the start of the thesis, the student presents the objectives of the work as well as some background on the topic. The presentation should not exceed 5 minutes.
- Thesis report:
  At the end of the project, no later than December 18, 2015, the student provides a written report to the supervisors (PDF via email suffices, no printing required). Together with the developed software, this report constitutes the main outcome of the project. Code should contain meaningful comments to allow for a follow-up project.
- Final presentation:
  After handing in the report, usually within two weeks after the end of the project, the student presents the achieved results. The expected time frame of the presentation is 15 minutes + 5 minutes questions.
- Evaluation of the work:
  The criteria for grading the work are described in [5].
- Finishing up:
  The used resources (e. g., laptop, desk, keys) should be cleaned up and handed back in. The complete material collected and developed throughout the thesis (design docs, code, thesis sources, etc.) must be committed to the provided subversion repository (svn://svn.ee.ethz.ch/tecstuds/rtrueb).

Further information about carrying out a thesis in our group can be found in [6].

References

[1] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[2] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, and D. Culler, "Elapsed time on arrival: a simple and versatile primitive for canonical time synchronisation services," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 1, pp. 239–251, July 2006.

[3] O. Saukh, D. Hasenfratz, and L. Thiele, "Reducing multi-hop calibration errors in large-scale mobile sensor networks," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, (New York, NY, USA), pp. 274–285, ACM, 2015.

[4] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: a testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proceedings of the 12th international Conference on Information processing in sensor networks (IPSN)*, 2013.

[5] TIK, "Notengebung bei Studien- und Diplomarbeiten." Computer Engineering and Networks Lab, ETH Zürich, Switzerland, May 1998.

[6] TIK, "Semester and Master Projects: Guidelines for Students and Advisors." Computer Engineering and Networks Lab, ETH Zürich, Switzerland, Apr. 2009.

Bibliography

[1] Tinyos home page. http://www.tinyos.net/. [accessed 10-December-2015].

[2] Tmote sky datasheet. http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf. [accessed 10-December-2015].

[3] R. J. Carroll, D. Ruppert, L. A. Stefanski, and C. M. Crainiceanu. *Measurement error in nonlinear models: a modern perspective; 2nd ed.* Monographs on Statistics and Applied Probability. Chapman and Hall, Boca Raton, FL, 2006.

[4] C. Frost and S. G. Thompson. Correcting for regression dilution bias: comparison of methods for a single predictor variable. *Journal of the Royal Statistical Society, Series A*, 163:173–189, 2000.

[5] C. Lenzen, P. Sommer, and R. Wattenhofer. Optimal clock synchronization in networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 225–238, New York, NY, USA, 2009. ACM.

[6] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, pages 153–166, New York, NY, USA, 2013. ACM.

[7] R. Lim, B. Maag, B. Dissler, J. Beutel, and L. Thiele. A Testbed for Fine-Grained Tracing of Time Sensitive Behavior in Wireless Sensor Networks. 2015.

[8] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 39–49, New York, NY, USA, 2004. ACM.

[9] O. Saukh, D. Hasenfratz, and L. Thiele. Reducing multi-hop calibration errors in large-scale mobile sensor networks. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, pages 274–285, New York, NY, USA, 2015. ACM.

[10] G. Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 1–12, New York, NY, USA, 2004. ACM.