Improving Reliability of IEEE-1588 in Substation Automation Based on Clock Drift Prediction

Yin Xiao
Abstract

Improving Reliability of IEEE-1588 in Substation Automation Based on Clock Drift Prediction

Yin Xiao

An electric substation is a node in the power grid network. It serves the purpose of transmitting and distributing electric energy from power sources to consumers. An electric substation is made of primary equipment and secondary equipment. The secondary equipment aims at protecting and controlling the primary one by sensing and analyzing various data. One pre-requisite to perform efficient protection functions is to have synchronized data provided by the various devices. The IEEE-1588 protocol is one promising way to handle the synchronized requirements of tomorrow’s substation automation, however, one of the remaining issue is its lack of reliability in case of the loss of the GPS signal (e.g., due to atmospheric disturbances or failure of the GPS antenna) which would lead to the de-synchronization of the devices inside a substation or between different substations.

The assignment of this master thesis project, commissioned by ABB CRC in Baden, is to investigate different drift clock prediction techniques which can handle the loss of the GPS signal, the loss of the GPS antenna receiver or the loss of the grand master device, thereby keep the substation automation synchronized without the GPS signal.

Various of linear and nonlinear models of time series prediction are explored in Matlab, five main approaches based on arithmetic average, weighted average and delay coordinate embedding are eventually chosen and developed in combination with an existing open source implementation of IEEE-1588 PTPd.

The five approaches' performance were judged and they have shown good results. Evaluation experiments run in our laboratory identify the most suitable technique for each type of GPS signal loss duration. On one hand, an arithmetic average based prediction technique can easily reach an accuracy of less than 10 microseconds for a prediction duration of a couple of seconds at a minimal computing cost. On the other hand, a time series-based prediction technique can provide an accuracy of 76 microseconds over a period of 48 hours but at a much higher computing power cost.

Keywords: IEEE-1588, Precision Time Protocol, Reliability, Substation Automation, Time Series, Prediction.
Contents

§ 1 Introduction .......................................................... 1
  1.1 Consideration and Background ................................... 1
  1.2 Problem Statement ................................................. 3
  1.3 Aims and Objectives .............................................. 7
  1.4 Dissertation Organization ....................................... 7

§ 2 State of the Art of Precision Clock Prediction ............... 9
  2.1 Introduction ........................................................ 9
  2.2 Overview of Physical Clock Model ............................ 9
  2.3 Time Series Prediction .......................................... 11
    2.3.1 Characterization of Time-Series ............................ 12
    2.3.2 Time-Series Analysis Techniques .......................... 15
  2.4 Precision Clock Prediction Using Time Series ............... 22
  2.5 Conclusion .......................................................... 23

§ 3 Fast Forecasting of Time-Series Data .......................... 25
  3.1 Introduction ........................................................ 25
  3.2 Arithmetic average and weighted average approaches ........ 26
  3.3 The Procedure of Time Series Prediction .................... 27
  3.4 Delay Coordinate Embedding .................................... 29
    3.4.1 Log Transformation ......................................... 34
    3.4.2 Auto-Regressive ............................................. 36
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Representation of a substation from a IEEE1588 point of view.</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Configuration of a substation after the failure of the master clock.</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>State machine of the prediction mechanism</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>The Procedure of Time Series Signals Prediction.</td>
<td>28</td>
</tr>
<tr>
<td>4.1</td>
<td>Structure of PTPd with its build-on</td>
<td>43</td>
</tr>
<tr>
<td>4.2</td>
<td>Message paths in the PTPd system</td>
<td>45</td>
</tr>
<tr>
<td>4.3</td>
<td>System diagram of clock servo</td>
<td>46</td>
</tr>
<tr>
<td>5.1</td>
<td>Samples of history Drift Data</td>
<td>53</td>
</tr>
<tr>
<td>5.2</td>
<td>The actual and predicted offsets time series using arithmetic average, weighted average and direct extrapolation</td>
<td>55</td>
</tr>
<tr>
<td>5.3</td>
<td>The actual and predicted offsets time series using AR model and log transform</td>
<td>56</td>
</tr>
<tr>
<td>5.4</td>
<td>Evolution of the (\theta) criteria for the five techniques over a varying horizon prediction</td>
<td>59</td>
</tr>
<tr>
<td>5.5</td>
<td>Evolution of the (\Theta) criteria for the five techniques over a varying horizon prediction</td>
<td>60</td>
</tr>
<tr>
<td>5.6</td>
<td>Evolution of Processing time for the five techniques over a varying horizon prediction</td>
<td>61</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Classification of Time Series ................................................. 12
2.2 Classification of Time Series Analysis Techniques ................. 21
2.3 Properties of Different Models ............................................. 22
1. Consideration and Background

An electric substation [1] is a node in the power grid network. It serves the purpose of transmitting and distributing electric energy from power sources to consumers, such as households or industrial plants. An electric substation is made of primary equipment (switchgears, breakers, transformers) and secondary equipment (sensors, merging units, intelligent electronic devices). The secondary equipment aims at protecting and controlling the primary one by sensing and analyzing various data.

One pre-requisite to perform efficient protection functions is to have synchronized data provided by the various devices. Depending on the considered function, the synchronization is either local, i.e., the devices of one substation are synchronized (e.g. busbar protection function) or global, i.e. the devices of two different substations are synchronized (e.g. line differential protection function). Moreover, the synchronization requirement ranges from 10 $\mu$sec to 100 $\mu$sec depending on the considered protection functions.
One promising way to handle the synchronized requirements in tomorrow’s substation automation is to use the *IEEE 1588 Precision Time Protocol* [2]. It enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The protocol is based on the master-slave paradigm in order to evaluate the relative offset and drift of each connected slave. One interesting feature of the protocol is its self-configurability which allows dynamically adding or removing any participating devices (either master or slave) by electing the best available clock at runtime.

From a time synchronization point of view, a typical substation automation system has the following architecture (figure 1.1). A GPS signal is received by a network device, called the grand master clock, inside the SA. The grand master clock transmits the GPS time using the *IEEE 1588* protocol to the devices (mostly IEDs, but also station PCs, gateways, transient fault recorders) connected to the network over TCP/IP.

In the outlined architecture, the GPS signal and its receiving part represent a single point of failure since the loss of the GPS means that the master clock will be running on its own and naturally drift away from the GPS clock. As an example, the consequences of atmospheric disturbances or thunderstorms hitting the GPS antenna imply the loss of the correct time and hence result in a desynchronization of different SA geographically distant which makes for example differential protection functions inoperable if the time offset across the SA becomes too large.
§ Chapter 1. Introduction

Figure 1.1: Representation of a substation from a IEEE1588 point of view.

1.2 Problem Statement

In the context of SA, the single point of failure characteristic of the architecture is problematic indeed. In the case of a transient failure\(^1\) of the GPS signal, a new master clock, called transient master, will be elected among the devices and will propagate its own time on the network (figure 1.2). Due to environmental conditions (e.g. temperature and pressure) or hardware imperfections (e.g., quartz quality), the transient master clock will drift from the GPS clock. Once the GPS signal comes back and the grand master clock takes over again, an offset will appear between the time sent by the GPS and the one present on the network. This can lead to a situation in which all devices will have to go through a re-synchronization stage. A re-synchronization stage implies an interruption of the control and protection functions running on the devices, as well as an abrupt

\(^1\)After a finite time the SA system will be back to the normal configuration
change of the devices’ time base which in turn may lead to a malfunctioning of the protection algorithms depending on time tagged data snapshots.

Although the version 2 of the IEEE 1588 standard introduces the concept of redundant master to improve the reliability of the protocol, the loss of the GPS signal due to atmospheric disturbances or the loss of GPS antenna are not addressed. Such scenarios, which are common in the case of an electric substation, must be handled by either keeping the substation synchronized to the GPS time or at least as close as possible to avoid a hard resynchronization of the clock’s devices and therefore a full re-initialization of the protection functions when the GPS signal is back.

Figure 1.2: Configuration of a substation after the failure of the master clock.

A clock drift prediction mechanism has been explored in this thesis project. The basic idea is to record on each device, or possible transient master, its offset and drift history during the normal configuration. Once a transient master is
§ Chapter 1. Introduction

elected, it then propagates its local time corrected with information gained from the offset and drift history in order to minimize the drift from the GPS clock. In the context of this approach, two different configurations are considered:

1. Normal configuration, where the GPS signal is received and propagated to the SA system.

2. Faulty configuration, where the GPS signal is missing, the grand master clock computer is down (temporary or permanent failure), or its network cable is unplugged.

During the normal configuration where the GPS signal is received and propagated to the SA system, each connected device, i.e., IEDs and the grand master clock, record its offset over several hours or days. Once a new device is elected as transient master in the faulty configuration, it performs the normal tasks accomplished by an IEEE1588 master. The difference is that every time it has to get its own current time, it will estimate its offset to the GPS clock and then correct the time which is sent to the slaves. The offset estimation is made out of the offset history, this can be done by, for example but not limited to, either computing an average offset or by identifying patterns out of the history data. The amount of recorded data for the offset history, as well as the prediction method used for the offset prediction, is a trade off between processing power, available resources and required time accuracy. In figure (1.3), each node can be either in a slave or transient master state. In a case of a slave state, the history is simply stored, while in a case of a transient master state, an estimated offset is calculated and then applied to the estimated real time.
Figure 1.3: State machine of the prediction mechanism
1.3 Aims and Objectives

The primary objective of this thesis project is to reduce and ideally avoid the offset between the GPS clock and the time known in a SA system while a GPS signal or the grandmaster clock is not available, in order to have a smooth re-integration of the GPS clock in the system. It therefore needs to design a time synchronization subsystem for SA which can handle the failure of a grand master clock for a given time frame (typically in the range of 48 hours). It needs the subsystem to handle three different disconnection scenarios, the loss of the GPS signal (e.g., in the case of atmospheric disturbances), the loss of the GPS antenna receiver (e.g., the antenna being hit by a thunderstorm) and the loss of the grand master device (e.g., hardware failure).

In order to achieve this goal, this master thesis investigates the possibility to use local history drift data to predict GPS time.

1.4 Dissertation Organization

Chapter 2 summarizes the state of the art of most commonly used linear and non-linear time-series predictions, introducing the ideas behind the various approaches that have been proposed in the past. It gives a definition of time series signals and some of their properties, it also classifies time series as well as methods for predicting their future.

Chapter 3 introduces five different approaches to clock drift prediction and discusses the detailed procedure of time series prediction. It then specializes the description to the autoregressive based delay coordinate embedding algorithm. It also introduces a logarithm transformation process to normalize the time series before model construction.
Chapter 4 provides the corresponding design descriptions of the prediction extension to an existing open source IEEE 1588 implementation, PTPd [3].

Chapter 5 evaluates the five different techniques with respect to their applicability in substation automation, and presents a quantitative comparison of those techniques in terms of processing power and prediction accuracy.

Chapter 6 rounds off the dissertation by summarizing the improvements made, and by offering suggestions for future work.

Appendix A shows the pseudo code of the implementation and the guideline for users.
2

State of the Art of Precision Clock Prediction

2.1 Introduction

In the present chapter, various precision clock prediction related techniques and algorithms will be introduced. First, a classical approach of modeling of clocks is outlined. Following, the definition of time series and its properties are introduced, the evolution of clock prediction is also presented. Finally, an overview over and a classification of time series as well as the most commonly used time series analysis techniques are provided.

2.2 Overview of Physical Clock Model

Normal clocks such as clocks at home and wristwatches usually drift compared to the actual time due to the influences of many factors. They often drift differently depending on their quality, the exact power they get from the battery, the sur-
rounding temperature, humidity, air pressure, and other environmental variables. Thus the same clock can have different clock drift rates at different occasions. A clock can be evaluated precisely only by comparison to other clocks. Therefore, evaluation of a clock actually refers to the measurement of the difference of two clocks. Normally, people conceptualize some of the laws of physics with time as the independent variable. However, in order to estimate the differences of two clocks, it is somehow crucial to have those laws inverted so that time is the dependent variable. In [4], it proposed a precise physical model for clocks and oscillators based on the fact that time as people now generates it is dependent upon defined origins, a defined resonance in the cesium atom, interrogating electronics, induced bias, timescale algorithms, and random perturbations from the ideal. In general, the reasons why does a clock deviate from others can be included within two categories. The first one is systematic deviation, i.e., frequency offset, time offset which are often environmentally induced; the second category is the random deviation which are usually not thought to be deterministic. Hence, at ideal level, to have a characterized clock model is one potential approach to exhaustively explore the relationship between the clock and GPS time. Nevertheless, [5] also demonstrates that it is complex to define such a model representing the quartz quality of a clock based on the influencing parameters. On the other hand, this will involve strict requirements of functional equipments by nature, e.g., multiple sensors are required in order to sense the conditional humidity, temperature values. However, this approach can not beforeseen in the context of today’s electric substation automations.

Characterizing the time-domain signal would become another option since it is viable to store the history drift values as a continuous or discrete data set by the device before GPS signal is lost. There are, obviously, numerous reasons to record data streams of time deviation between a clock and some primary reference.
Among these is the wish to gain a better understanding of the underlying context of the drift generating mechanism from the history drift record, the prediction of future deviations based on past measurements of drift, or the optimal control of the system.

### 2.3 Time Series Prediction

From the point of view of time series, the observations collected are results of the state of the system. A time series is defined as a collection of measurements of a variable that are typically taken at successive times, and usually spaced at equal time intervals. This data set can be decomposed into components including trends, seasonality, and random noise [6]. A trend in most time series patterns represents a general systematic linear or nonlinear component that changes over time and does not repeat or at least not repeat within the time range captured by the sample data, and the seasonality refers to patterns that are recurring in correspondence to the date, it may have a formally similar nature as a trend, however, it repeats itself in systematic intervals over time. It is also assumed that a time series consist of systematic pattern (a set of identifiable components) and random noise which can be caused by errors in the observation methods. Random noise typically makes the pattern difficult to identify. The characteristic property of a time series is the fact that the data are not generated independently, their dispersion varies in time, they are often governed by a trend and they have cyclic components. Time series are assumed to have been generated by dynamic systems, it is therefore important to know when the measurements were taken, i.e., it must either be assumed that the measurement signal has been equidistantly sampled, or alternatively, the time instant when each sample was taken must be stored together with the time series as a second piece of information.
In [4], it described a general systematic model for clocks and oscillators which is defined by the following expression:

$$\Delta t = a_0 + a_1 t + U(t) \quad \text{(2.1)}$$

where $\Delta t$ is the time offset measured by comparison with the reference clock, $t$ the time elapsed after the last synchronization, $a_0$ the synchronization error or clock bias, $a_1$ the clock drift, that is, the relative frequency offset of the clock compared with the reference clock, and $U(t)$ the random error, which depends on the noise spectrum of the clock. As discussed before, the random error $U(t)$ consists of various of influencing parameters which are often environmentally induced can not be measured since deployment of sensors is not viable, it is consequential to concentrate on the offset part, $a_1$, i.e., time series of history drift data.

### 2.3.1 Characterization of Time-Series

In [7], it provided a useful classification of different time series, [8] also enhanced the classification by adding several detailed classes, in table (2.1), it is repeated with a few abatements.

<table>
<thead>
<tr>
<th>1</th>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Natural</td>
<td>Synthetic</td>
</tr>
<tr>
<td>3</td>
<td>Deterministic</td>
<td>Stochastic</td>
</tr>
<tr>
<td>4</td>
<td>Stationary</td>
<td>Non-stationary</td>
</tr>
<tr>
<td>5</td>
<td>Linear</td>
<td>Non-linear</td>
</tr>
<tr>
<td>6</td>
<td>Short</td>
<td>Long</td>
</tr>
<tr>
<td>7</td>
<td>Single Recording</td>
<td>Multiple Recordings</td>
</tr>
</tbody>
</table>
The system from which the time series is drawn can be either discrete or continuous. A discrete time series is one where the set of times at which observations are made is a discrete set. Continuous time series are obtained by recording observations continuously over some time interval.

In academia, time series are quite often synthesized from simulation experiments. Techniques that are capable of dealing with such synthetic time series are not necessarily also well suited to deal with natural (measured) time series. Hence it is important to record whether a time series is natural or synthetic.

If a time series can be exactly predicted from past knowledge, it is termed as deterministic, i.e., there exists a scalar function \( f \) which can express dependence between two quantities. One of which is given (the independent variables, e.g., the previous observations) and the other produced (dependent variable, i.e., the estimation). Otherwise it is termed as statistical, where past knowledge can only indicate the probabilistic structure of future behaviour. A statistical series can be considered as a single realisation of some stochastic process. A stochastic process is a family of random variables defined on a probability space.

Many of modern theory of time series, especially statistical techniques, rely on stationary process, if a time series is non-stationary, they might be unsuitable of dealing with it. For this reason time-series analysis often requires one to transform a non-stationary series into a stationary one so as to use this theory, for instance, regular differencing and seasonal differencing might be used for removing the trend and seasonality, respectively. From an intuitive point of view, a time series is said to be stationary if there is no systematic change in mean (no trend), if there is no systematic change in variance and if strictly periodic variations have been removed [9].

The system from which the time series is drawn may be either linear or non-
linear. A linear system is a mathematical model of a system based on the use of a linear operator, they typically exhibit features and properties that are much simpler than the nonlinear case. Non-linear process is always being exploited by some of the computational intensive techniques.

Time series can be either short or long. A short time series may be caused by a transitory event, such as a surgery, that is of limited duration. Although it is possible to increase the sampling rate, thereby making the time series longer, this may not help, e.g., it is pointless to make the time series longer by shorten the synchronization round. There exists a natural sampling rate for each time series that is related to the natural frequencies (eigenfrequencies) of the system from which the time series is sampled. Oversampling increases the length of the time series, but not its information contents. Forecasting techniques that are based on models suffer from data deprivation in the presence of short or oversampled time series. The simple extrapolation techniques may work best in this case, at least for single time series. Multiple correlated time series may still be better predicted using model based approaches.

A time series may consist of a single recording, or of multiple recordings. Multiple recordings lead to multiple uncorrelated trajectories representing different patterns of the same phenomenon. However, from an intuitive point of view, it seems to be difficult to evaluate if it is the same phenomenon as observed before or not by only judging the records of the evolution of a single variable, drift value.

Finally, time series may be documented or blind. These terms relate to the amount of knowledge available about the systems from which the time series was drawn. Obviously, such knowledge can be exploited exhaustively by prediction approaches, especially deductive ones, and make use of such knowledge to infer a model structure that matches that of the underlying system.
2.3.2 Time-Series Analysis Techniques

Unlike the analyses of random samples of observations that are discussed in the context of most other statistics, the analysis of time series is based on the assumption that successive values in the data file represent consecutive measurements taken at equally spaced time intervals, it accounts for the fact that data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for. Time series analysis concerns itself with the investigation of single or multiple observations of measurement data streams taken from a system under observation, i.e., extract data that maximally informative for system model constructing purpose. It is a characteristic property of time series that they never contain complete information about the system being observed, and in particular, that the excitations that are imposed on the system are not under the observer’s control, and are, in many cases, unknown to them.

The desire to predict the future and understand the past drives the search for laws that explain the behavior of observed phenomena; Examples range from the volatility of financial markets to the irregularity in a heartbeat. If there are known underlying deterministic equations, in principle they can be solved to forecast the outcome of an experiment based on the knowledge of the initial conditions. To make a forecast if the equations are not known, one must find both the rules governing system evolution and the actual state of the system. The relationship between observed phenomena of a system or the knowledge of its properties is termed as the model of the system, therefore, the aim of modeling is to find a description that accurately captures features of the long-term behavior of the system while the aim of forecasting (predicting) is to accurately predict the short-term evolution of the system. Generally, there are two potential roads lead to

\[1\) It is worth noting that these two are not necessarily identical.
predictions:

1. predicting was done by directly extrapolating the series through a global fit in the time domain.

2. attempts to create a model that explains relationships between observations made in the past, and then fit the model in a simulation to make predictions of future.

Evolution of Time Series Prediction

In the early days, prediction methods are simple techniques that do not rely on training a system model first, but directly use the available data to make predictions about the future, namely extrapolation, the previously made predictions make no effect on further predictions. Direct extrapolation is inherently unsafe since it does not provide for means that would allow to estimate the quality of predictions made [8], i.e., estimate the error associated with any prediction made. Modeling is a better approach, it enables operations on input variables, thereby allowing users to formulate various of scenarios and observe the consequences that might result when implementing any one of those scenarios.

Historically, the general theory of first modern time series approaches can be traced back to [10] [11], it was a linear model which allows to correlate different observations with each other, thereby improving the potential of making good predictions. It has three particularly desirable features:

1. can be used to represent both stationary and non-stationary processes.

2. straightforward to implement.

3. a requiring reasonable computation time.
The penalty for this convenience is that they may be entirely inappropriate of dealing with non-linear behavioral systems since they always assume the systems from which the time series has been measured to be linear and to operate under stationary conditions [12].

In order to have time series analysis widely used in systems especially with non-linear characteristics, non-linear models were proposed in [13]; Moreover, in order to be able to deal time series that exhibit non-stationary characteristics, in [14] [15], pre-filtering methods were developed that remove the effects induced by trend or seasonality of non-stationary time series. However, the choice of linear or non-linear ones is a tradeoff in terms of accuracy and processing overhead, moreover, the later ones need more training data (observations) and are less well behaved. In case of short synchronization round of IEEE1588, non-linear predictors combined to the other tasks running on a node may not have enough time to proceed.

Some of other important contributions were the construction and identification of state-space models [16], the introduction of learning techniques for model identification [7]. The use of learning techniques constitutes an important trend in modern prediction approaches. it make less structural assumptions about the system from which the variables were generated, and that are therefore more generally applicable to a wider class of prediction problems.

The most widely used among these is neural network, it has been successfully applied to financial markets prediction [17] [18] and wind power forecasting [19] [20], etc. Although previous comparisons of neural networks and linear predictors have shown that neural network sometimes can give better results [21] [22], it meanwhile requires additional procedures (e.g., genetic computing\textsuperscript{2}) and much more computationally expensive training cost.

\textsuperscript{2}A method used to construct neural networks architecture
There exists a vast literature on methods for analyzing time series. There even were organized competitions on a worldwide level in order to advance the state of the art of methodologies for time-series analysis and prediction [23]. Time-series analysis has been applied to many different application areas, such as the prediction of financial markets (economical models), or the monitoring of physiological signals stemming from a patient during surgery (biomedical systems). In engineering, time-series analysis is of interest in the contexts of instrumentation and filtering of signals, and the design of predictive controllers. In particular, time-series analysis also has been used by many researchers for precision clock prediction.

When the underlying dynamical system is not fully understood, observed regularities in the time series can provide insights into the inner working of the system. Given a time series from the past, the aim of a prediction algorithm is to forecast the expected future of the time series by drawing analogies from previous behaviour, and thus, indirectly forecast the future behaviour of the system.

Classification of Time-Series Analysis Techniques

A first coarse classification can be made by distinguishing between prediction and simulation approaches, i.e., techniques that operate on the time series directly versus techniques that create a model in advance and then operate on that model. Yet, this classification is not truly crisp. Even simple extrapolation techniques usually identify parameters of a polynomial or regression function. Whether this polynomial or regression function is called a model is simply a question of taste.

A second classification can be made by distinguishing between deductive and inductive modeling techniques. Again, also this classification is not strictly crisp. In fact, there are no strictly deductive approaches to time-series analysis, even in the case of an extremely well documented time series [8]. For instance the
DVS algorithm described in [7], a deductive modeling approach would make use of the knowledge provided about the system to conclude that the output signal is governed by the linear equations, a set of $k$ very simply and well understood linear equations that are furthermore autonomous. The modeler would thus only need to identify the parameters of the equations, such that they match the observed output patterns optimally (Least square method). The technique is almost purely deductive, except for the identification of the $k$ nearest neighbors, which can be considered an inductive process. In general, the more structural assumptions are being made about the system from which the time series is drawn, the more the approach must be considered deductive.

Among the various available primarily inductive modeling approaches, the following two: Artificial Neural Networks (ANN) [24] and NARMA [25] are the most classical ones. They can better deal with non-stationary behavior since ANNs and NARMA do not exploit stationary explicitly, except training the weights of neurons may become more problematic in the case of non-stationary time series. As described above, they are not viable approaches in the context of substation automations.

Based on the previous discussion, the choice of deductive techniques in this work can be narrowed down to the choice of linear processes. Time series analysis produced by linear predictors is based on the fact that a finite-dimensional linear system produces a signal characterized by a finite number of frequencies. There exist highly successful methods of time series prediction based on the exploitation of this fact in the time domain also frequency domain. Such as Auto Regressive (AR), Moving Average (MA) and ARMA. Based upon Allan’s general clock model, linear techniques were employed by most of investigators [26] for the prediction of precision clocks.

Moving average (MA) works by calculating the average of a small set of past
data in which each data subset average is calculated, it describes the time series as a moving average of white noise. The MA method operates in an open loop without feedback, it can only transform an input that is applied to it and limited to stationary processes, or those processes that do not have trend or periodic fluctuations and do have an unvarying variance over time. Some feedback provided are necessary for generating internal dynamics of a system, this would lead to another model, namely autoregressive.

One of the first linear autoregressive (AR) models was proposed by [27] for sunspots study. This method can be used to represent non-stationary processes as well as stationary processes. In his model, it predicted the next value as a weighted sum of previous observations of the series, the value of the series at the current time is as function of the previous values added to the random variation term which can represent either a controlled input to the system or noise depending on the application. The desirable objective of training an autoregressive model consists of setting each weight to an optimized value, i.e., gives the best prediction in the sense that that model minimizes the squared error within the model class. However, by comparing with MA models, AR models give limited freedom in describing the properties of the disturbance terms.

Furthermore, processes may be a combination of moving average and autoregressive process which require the ARMA model. The advantages of ARMA include a requiring reasonable computation time and creation of a tool for analysis, forecasting, and control. One restriction of the ARMA model is that it was not designed for time series with asymmetry or data with sudden bursts of large amplitude at irregular times [28]. In addition, ARMA is used with the assumption that the underlying system is linear. The AR method is useful because it only requires knowledge of the system’s output values and can be used for both stationary and non-stationary time series.
In conclusion, the most important advantages of AR models are its ability to generate decent models of time series in a fairly automated manner, its relatively less computational overhead, its flexibility to fit both stationary and non-stationary system, its firm mathematical basis [29], finally, less knowledge it requires from the system. Table(2.2) presents the overview of the pros and cons of these classification proposed in this chapter.

Auto-Regressive model mixed with a arithmetic approach and a weighted average approach shall be used in the subsequent chapters of this thesis, although only AR shall be discussed in any great detail. For the reasons discussed above, it makes sense to use these relatively less tricky approaches if given only the history data set of drift taken in sequence which is not informative enough for other approaches. The strict requirements of processing cost is the other important consideration as well.

Table 2.2: Classification of Time Series Analysis Techniques

<table>
<thead>
<tr>
<th>Type of Time series</th>
<th>Type of Modeling Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AR</td>
</tr>
<tr>
<td>Continuous</td>
<td>**</td>
</tr>
<tr>
<td>Discrete</td>
<td>**</td>
</tr>
<tr>
<td>Deterministic</td>
<td>**</td>
</tr>
<tr>
<td>Stochastic</td>
<td>*</td>
</tr>
<tr>
<td>Stationary</td>
<td>**</td>
</tr>
<tr>
<td>Non-stationary</td>
<td>*</td>
</tr>
<tr>
<td>Linear</td>
<td>**</td>
</tr>
<tr>
<td>Non-linear</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 2.3: Properties of Different Models

<table>
<thead>
<tr>
<th>Properties of Models</th>
<th>Type of Modeling Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AR</td>
</tr>
<tr>
<td>Wide Usability</td>
<td>*</td>
</tr>
<tr>
<td>Coding Complexity</td>
<td>*</td>
</tr>
<tr>
<td>Input Information</td>
<td>*</td>
</tr>
<tr>
<td>Computation Cost</td>
<td>*</td>
</tr>
</tbody>
</table>

2.4 Precision Clock Prediction Using Time Series

The last three decades has seen some decisive contributions in precision clock prediction field, made possible as a consequence of the appearance of creating various of systematic or random models based on Allan Variance\(^3\) (also known as two-sample variance) that allowed to characterize the systematic and random deviations of clocks. Important systematic deviations may include information of daily and annual dependences which can be manifestations of environmental effects such as deviations induced by humidity, temperature and pressure, it opens the door to optimum estimation of precision clocks.

Based upon the theory in [4], the choice of predictors can be confined considerably to linear ones since they are sufficient enough to optimally predict clocks whose stability is described by an extension of the common oscillator model, including drifts [26]. For the two decades following Allan, the reigning paradigm remained that of linear models driven by noise [30] [31]. In order to make future predictions better than the straightforward linear ones, lots of optimum predic-

\(^3\)Allan Variance is a method invented by David W. Allan for measuring stability in a variety of exotic precision clocks
tors in terms of accuracy came out [32] [33], these new developments were mainly focused on the noise modeling since the optimum precise-clock prediction is somehow equivalent to optimum precise-clock noise prediction. Again, the penalty for these potential improvements is the heavier processing load.

On the other hand, there also exist some researchers strived towards less computation overhead in order to meet the requirements of potential real time applications. [34] described a fast, easily coded numerical algorithm (which is competitive with conventional k-nearest-neighbor approaches) for convenient initial analysis of time series data. The algorithm [26] proposed was for the generation of time scales where the best predictor is chosen by comparing the value of various standard deviations in real time. It is simpler and faster than iterating to create linear models in real time using least square method. However, in that case, a table consisted of a bunch of predictors as well as the related performance calculated under different scenarios should be stored by the device, that means it involves a set of experiments have to be run in advance on every substation or every node which is susceptible to perform such operations.

2.5 Conclusion

This chapter provided an introduction to the nature and importance of time series analysis techniques. It was defined what a time series is and what are its basic properties, it also offered an overview over and a classification of the different types of time series to be found. The most commonly linear and neural networks models were outlined. It finally suggested the commonly used deductive modeling approaches for precision clock prediction. This thesis shall mainly deal with the linear autoregressive model, the proposed methods are all based on sequences observed from single source used as training data set, and normally, the training
§ Chapter 2. State of the Art of Precision Clock Prediction

data set is a continuous sequence of drifts recorded naturally before loss of GPS signal.

There also exist other two naive techniques, namely arithmetic average method and weighted average method which are not belong to modeling techniques. However, they will, within this work (especially short-term time horizon), surprisingly be useful. A fact of these two will be extensively discussed and exploited in subsequent chapters of this thesis.
3

Fast Forecasting of Time-Series Data

3.1 Introduction

Already in Chapter 2, it was mentioned that one of the primary goals of this work is to gain a suitable way to get devices within one substation or geographically distant substations synchronized in case of the lose of a reference clock (e.g., GPS signal). In this Chapter, the application of three different approaches (a arithmetic average, a weighted average and Delay Coordinate Embedding algorithm) to clock drift prediction are presented.

In this thesis, these predictors strictly operate on naturally recorded uni-variate time series. i.e., the arithmetic average and weighted average ones directly extrapolate the past drifts while the Delay Coordinate Embedding approach analyzes observed patterns of measurement drifts, and predicts the future behavior on the basis of its own past, without ever identifying the systems status under which these signals were generated. These methods perfectly fit the realistic situation that lack a large number of free parameters, unlike such methods as Artificial Neural Networks, which leads to enhanced time requirements, and perhaps more strict
requirements on input knowledge.

### 3.2 Arithmetic average and weighted average approaches

The arithmetic average approach simply calculates the average over the last \( n \) synchronization rounds, i.e., the first estimation is computed by averaging over the last \( n \) observations, and then iterates to get the next estimations by shifting the previous \( n \) observations until the GPS signal is back. Thus, the arithmetic average approach takes the accumulation strategy which utilizes the new estimations as parts of the history data in the next round, as does the weighted average approach.

\[
\hat{x}_t = \frac{1}{t-1} \sum_{i=1}^{t-1} x_{t-i}
\]

It is not surprising that arithmetic average calculation performs well if the clock drift is stationary, especially in short-term predictions. The most important advantage of doing average would be the negligible processing cost. It is worth noting that, due to the low processing overhead, each node of a substation can run its own arithmetic average prediction. Such a scenario will allow each node to handle the loss of one or two consecutive IEEE 1588 synchronization messages due to networking problems.

The weighted average approach is similar to the arithmetic average one but only takes into account the drift values of which the probability is greater than a given threshold value over the last \( n \) rounds. For instance, select a group of most frequent ones (i.e., partially filter the possible noises), then perform the weighted average over that group.
\[ x_t = \sum_{i=1}^{t-1} \alpha_{t-i} \ast x_{t-i} ; \quad (\alpha_{t-i} \geq \vartheta) \]

where \( \alpha_{t-i} \) represents the probability of element \( x_{t-i} \) during last \( n \) observations, \( \vartheta \) the given threshold value of filter.

### 3.3 The Procedure of Time Series Prediction

The prediction of time series signal is based on their past values. Therefore, it is necessary to obtain a data record. When obtaining a data record, the objective is to have data that are maximally informative and an adequate number of records for prediction purposes. Hence, future values of a time series \( x_t \) can be predicted as a function of past values \( x_{t-1}, x_{t-2}, \ldots, x_{t-m} \).

\[ x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-m}) \]

The problem of time series prediction now becomes a problem of system identification. The unknown system to be identified is the function \( f(\cdot) \) whose inputs are the past values of the time series. While observing a system there is a need for a concept that defines how its variables relate to each other. As described in Chapter 2, the relationship between observations of a system or the knowledge of its properties is termed as the time series model, hence, Equation (3.1) can be called a model of the system.

The search for the most suitable model for a system is guided by an assessment criterion of the goodness of a model. In the prediction of time series, the assessment of the goodness of a model is based upon the prediction error of the specific model. Moreover, in this work, processing cost and the amount of free input parameters required are other two criterions of equal importance.
Figure 3.1: The Procedure of Time Series Signals Prediction.
After the most suitable model of a system has been determined, it has to be validated. The validation step in the system identification procedure is very important because in the model identification step, the most suitable model obtained was chosen among the predefined candidate models. This step will certify that the model obtained describes the true system. Usually, a different set of data than the one used during the identification of the model, the validation set, is used during this step.

3.4 Delay Coordinate Embedding

In brief, the delay-coordinate-embedding approaches identify the state of the system at certain time points, search the past history of observations for similar states, and, by studying the evolution of the observable outputs following the similar states, infer information about the future of the system. As well as other classical prediction techniques, for a time series \( \{x_i\} = x_1, x_2, ..., x_N \), the delay-coordinate-embedding approach attempts to fit models of the following form, but with a linear function \( f(\cdot) \).

\[
x_{i+T} \approx f(x_i, x_{i-\tau}, ..., x_{i-(m-1)\tau})
\]

The integers \( T \) and \( m \) define the following quantities:

1. \( T \): lead time or look ahead time, (i.e., prediction time into the future).

2. \( m \): embedding dimension or resolution, (i.e., the degree to which information about individual sequence elements is preserved).

Furthermore, the \( m \) past values are combined in the delay vector \( X_i \) in which the elements are time delayed data values from the time series. The spacing of the
samples of the delay line is equal, i.e., $X_i := (x_i, x_{i−\tau}, \ldots, x_{i−(m−1)\tau})$, where $\tau$ is the lag time between each sample, it will be made equal to 1 (i.e., 2 seconds) here, since the time series collected is already sampled fairly coarsely.

The basic idea of the delay-coordinate-embedding approach is prediction of future values given the current delay coordinate vector. In this method the time series (history drift sequence) is broken up into equal-sized chunks which can indicate the internal state of the system, i.e., reflect the evolution of the system. Whenever a prediction needs to be made, the latest chunk is found, and the chunks nearest to it in the chunks space are retrieved. By observing how the time series developed in each of these cases, its expected course under the current circumstances can be predicted. This involves several issues:

1. Decide the size of the chunks.
2. Decide the number $k$ be used.
3. Decide the method of interpolation for combining the results from the $k$ nearest subsets found.

In our experiment, the size of the chunks were chosen on an ad hoc basis, i.e., having a bunch of experiments applied on the same data set by varying the embedding dimension systematically. The choices of number of nearest neighbors are also based on practical applications, i.e., if $k$ is too small, sensitive to noise points, otherwise, neighborhoods may include points which are not that close to the latest

---

1. Time series $(x_t, x_{t−1}, \ldots, x_1)$ can be broken up into several chunks like $(x_t, x_{t−1}, \ldots, x_{t−i}), (x_{t−1}, x_{t−2}, \ldots, x_{t−i−1}), \ldots, (x_{t+i+1}, x_{t+1}, \ldots, x_1)$. These all chunks compose a so-called chunk space.
2. To predict $x_t$, the chunk which has $x_{t−1}$ as the last element is the latest chunk, e.g., $(x_{t−1}, x_{t−2}, \ldots, x_{t−m})$. 
Chapter 3. Fast Forecasting of Time-Series Data

chunk. The *Euclidean Distance* was used for identifying neighborhoods. For solving the third problem, the \( k \) nearest neighbors \( X_i^1, X_i^2, \ldots, X_i^k \) of the current state vector \( X_i \) are found, and the time series values immediately succeeding these \( k \) instance (denoted \( p_1, \ldots, p_k \)) are used to estimate the value after \( X_i \). One method is to perform arithmetic average over \( p_1, \ldots, p_k \), another more tricky one would be utilizing an Auto-Regressive model:

\[
x_t = \sum_{i=1}^{m} \beta_i \cdot x_{t-i} + \epsilon_t
\]

which uses the least squares method to find the linear function \( f : IR^m \rightarrow IR \) that gives the best prediction for \( x_{i+1} \) in the sense that it minimizes the squared error within the model class. On average, the Gaussian white noise \( \epsilon \) is assumed to be small relative to \( x_t \), therefore, \( \hat{x}_t \) can be estimated by using:

\[
x_t \approx \hat{x}_t = x_t - \epsilon_t = \sum_{i=1}^{m} \beta_i \cdot x_{t-i}
\]

One particular implementation of delay-coordinate-embedding approach is given by [16]. Here we describe it but with some modifications\(^3\) which are compliant to the scope of this thesis work.

1. Data pre-processing. A function that maps the entire time series to a new set of replacement values such that each old value can be identified with one of the new values, i.e., using a natural logarithm transformation.

2. Divide the time series into two parts:

   (a) a training set \( x_1, \ldots, x_{N_f} \) used to estimate the coefficients of each model.

\(^3\)The algorithm presented here differs from [16] in that we update the training data after each new estimation is made rather than holding it fixed.
(b) a test set \( x_{N_f+1}, \ldots, x_{N_f+N_t} \) used to evaluate the model. \( N_f \) denotes the number of points in the training set, \( N_t \) denotes the number of points in the test set.

3. Choose \( m, k \). (outer loops)

4. Choose an input signal \( X_i \) (i.e., test delay) for a forecasting task \( (i > N_f - 1) \).

5. Compute the distances \( d_{ij} \) of the test vector \( X_i \) from the training vectors \( X_j \) (for all \( j \) such that \( (m - 1)\tau < j < i - 1 \)).

6. Order the distance \( d_{ij} \) (Using the quick sort algorithm).

7. Find the \( k \) nearest neighbors \( X_j^{(1)} \) through \( X_j^{(k)} \) of \( X_i \), and fit an affine model with coefficients \( \beta_1, \ldots, \beta_m \) of the following form:

\[
x_i^{(l)} \approx \sum_{i=1}^{m} \beta_i \ast x_{i-l}^{(l)}, (l = 1, \ldots, k.) \tag{3.7}
\]

In time series context, this is an Auto-Regressive model of order \( m \) fitted to the \( k \) nearest neighbors to the test points; i.e., there are \( k \) equations. \( j \) denotes those times in the training set where the dynamics is most similar to the test point.

8. Use the fitted model from step (7) to estimate a one-step-ahead forecast \( \hat{x}_i(k) \) starting from the test vector, and compute its error:

\[
e_i(k) = \hat{x}_i - x_i \tag{3.8}
\]
9. Repeat step (4) through (8) as (i+1) runs through the test data, but replace the \( x_i \) by the lastest prediction, \( \hat{x}_i \), i.e., instead of having \( x_{i+1}, x_i, \ldots, x_{i-(m-2)} \), now, \( \hat{x}_{i+1}, x_i, \ldots, x_{i-(m-2)} \) is the input used to fit the model for next estimation. The final offset \( \theta \) and possible maximum offset \( \Theta \) can be computed as:

\[
\theta_m(k) = \sum_{i=1}^{N_t} e_i(k) \tag{3.9}
\]

\[
\Theta_m(k) = \text{Max} \theta_m(k) = \sum_{i=1}^{t} e_i(k), \ t \in [1, N_t] \tag{3.10}
\]

The Delay-Coordinate-Embedding algorithm as presented here uses a **local linear approximation** in step 7. The idea of systematically varying embedding dimension and neighborhood size is not restricted to local linear models. Also, the model can be made considerably more detailed by introducing a variety of noise terms, as is typically done in non-stationary modeling [28]. Furthermore, in step 9, to get an estimate for \( x_{t+1} \), there are two obvious choices. The **direct prediction** method means that the original method is applied to \( x_{t-i}, \ldots, x_{t-1} \) to predict two times units ahead. In contrast, **iterated prediction** means applying the method to \( x_{t-i}, \ldots, x_{t-1}, \hat{x}_t \) to predict one unit ahead. The direct prediction only uses real observations for the forecast, whereas the iterated prediction also uses previously made forecasts as if they were real observations. Much discussion has ensued over which choice is superior. The reliability of direct prediction is suspect because it is forced to predict farther ahead. On the other hand, iterated prediction uses \( \hat{x}_t \), which is possibly corrupted data. However, [35] argue that iterated prediction is superior.
3.4.1 Log Transformation

A natural logarithm transformation can be applied in step(1) of the delay coordinate embedding algorithm, prior to creating the linear predictor based on the local approximation technique [36]. It is actually a concept of data preprocessing which attempts to stabilize the variance of the time series. It has usually been tried when sometimes one is not very clear what kind of variance variability the time series has [37]. The logarithm transformation may reduce the effects introduced by noises (spikes), the detailed reasons why it is decent for this research are given below:

1. It is believed that the reason why it was so good is that it tended to give the same output as a linear predictor when the drifts were about constant. This was because the log predictor had approximately the same weights as the linear predictor, and averaging in log space is the same as averaging in linear space if the points being averaged are close together.

2. On the other hand, in the event of spikes (as occurred in a small part of the data set) the log predictor gave much better prediction results. This may reflect the geometric averaging process as being better than the arithmetic averaging process following a spike.

To be more specific, the second reason can be explained by presenting the output pattern of linear regressive

\[ \hat{x}_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2}, \ldots, + \alpha_m x_{t-m} \] \hspace{1cm} (3.11)

which can be assumably considered as an arithmetic average if the elements of coefficients vector \( \alpha \) have the equal values, i.e., \( \alpha_1 = \alpha_2 = \ldots = \alpha_m \).
\[ \hat{x}_t = \frac{x_{t-1} + x_{t-2}, ..., + x_{t-m}}{m} \]  
(3.12)

By carrying out the natural logarithm transformation in advance, the output pattern would be transformed to

\[ \hat{x}_t = \alpha_1 \log x_{t-1} + \alpha_2 \log x_{t-2}, ..., + \alpha_m \log x_{t-m} \]  
(3.13)

which especially can be written in short

\[ \log(\hat{x}_t) = \log(x_{t-1} \cdot x_{t-2}, ..., \cdot x_{t-m})^{\frac{1}{m}} \]  
(3.14)

if \( \alpha_1 = \alpha_2 =, ..., = \alpha_m \). Therefore, the prediction can be defined by the following:

\[ \hat{x}_t = \sqrt[m]{(x_{t-1} \cdot x_{t-2}, ..., \cdot x_{t-m})} \]  
(3.15)

which can be considered as a geometric averaging. Therefore, in case there exists spike(s)\(^4\) within the the training data, the estimation produced by geometric averaging process would be closer to the expectation value of that sequence than the arithmetic one, for instance given a time series \( X_i \)

\[ X_i = (9, 11, 2, 10) \]  
(3.16)

the arithmetic average of \( X_i \) is

\[ \frac{9 + 11 + 50 + 10}{4} = 20 \]  
(3.17)

\(^4\)Spikes here refer to the history points which are not frequently occurs within a certain period
while the geometric average one is

$$\sqrt[3]{9 \cdot 11 \cdot 50 \cdot 10} \approx 14.915 \quad (3.18)$$

Generally, the probability for the succeeding time series which arranges from 9 to 11 should be relatively higher since the third element 50 is considered as a spike of this time series, therefore, logarithm transformation works well.

### 3.4.2 Auto-Regressive

Linear autoregressive is one of the models that has been used in step 7 of procedure delay-coordinate-embedding. It is a classical approach to combine the results from the $k$ nearest neighbors found. The general form of the AR model is given by the linear equation:

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2}, ..., + \alpha_m x_{t-m} + \epsilon_t \quad (3.19)$$

$$x_t \equiv \hat{x}_t + \epsilon_t \quad (3.20)$$

where the current value of the time series is expressed as a weighted sum of past values plus the white-noise term $\epsilon_t$. Thus, $x_t$ can be considered to be regressed on the $m$ previous values of $x(t)$. If on average $\epsilon_t$ is small relative to $x_t$, the Gaussian white noise can be neglected. To be more specific, calculating $\hat{x}_t$ is therefore a matter of evaluating the coefficients $\alpha_1, ..., \alpha_m$. In the context of drift clock prediction, the previous linear equation is transformed as a matrix calculation as follows:
Chapter 3. Fast Forecasting of Time-Series Data

\[ \hat{x}_t = [x_{t-1} \ x_{t-2} \ \cdots \ x_{t-m}] \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix} \]

the \( \alpha \) coefficients are then evaluated from the following matrixal equation:

\[
\begin{bmatrix} x_{t-1} \\ x_{t-2} \\ \vdots \\ x_{t-k} \end{bmatrix} = \begin{bmatrix} x_{t-1-1} & x_{t-2-1} & \cdots & x_{t-m-1} \\ x_{t-2-1} & x_{t-3-1} & \cdots & x_{t-(m+1)-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t-k-1} & x_{t-(k+1)-1} & \cdots & x_{t-(m+k-1)-1} \end{bmatrix} \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix}
\]

or in short:

\[
A\alpha = b \tag{3.21}
\]

If \( m = k \), the matrix equation represents a set of \( k \) linear equations in \( m \) unknowns that can be solved in an unique fashion using any technique suitable for solving linear systems of equations. If \( m < k \), it represents an overdetermined set of linear equations in \( m \) unknowns that can be solved approximately in a least squares sense which can be interpreted as a method of fitting data. The best-fitting in the least-squares sense is that instance of the model for which the sum of squared deviations from a given set of data has its least value. A deviation being the difference between an observed value and the output given by the model. The linear regression can be achieved by multiplying the transposed matrix \( A^T \) at both sides of equation as:
\[ A^T A \alpha = A^T b \] 

and hence

\[ \alpha = (A^T A)^{-1} A^T b \]

is an approximate solution of the set of equations, where \((A^T A)^{-1} A^T\) is a pseudo-inverse of matrix \(A\).

Once the coefficient vector \(\alpha\) has been found, future estimations can recursively be obtained using the equation:

\[ \hat{x}_t = \alpha X_{t-1} \]

where:

\[ X_{t-1} = \begin{bmatrix} x_{t-1} & x_{t-2} & \cdots & x_{t-m} \end{bmatrix} \]

this concludes the straightforward description of the method.

It is of interest to discuss the stability of an AR model. If the uni-variate time series is stationary, it seems reasonable to expect that recursive predictions produce a stationary forecast as well. However, there is no guarantee that the least squares approach to determining the parameter values of the AR model will satisfy the stability requirement, i.e., recursive predictions using Eq.(3.16) may grow beyond all bounds. One promising approach to ensure the stability of AR model is to request that:

\[ \sum_{i=1}^{m} \alpha_i = 1.0 \]
by making use of autocorrelation function. However, it will cause a computational intensive training process since expected values and variances of time series need to be calculated through the vector $X_{t-1}$.

3.5 Model Validation

The last step in the system identification procedure is to validate the identified model. Validation of a model can be performed in a number of ways. A possible validating method is to check whether the model satisfies its purpose. For example, evaluating the prediction error over a validation data set that is different from the data set used during the identification of the model. A very useful technique for validation of a model is the residual analysis. The residuals carry information about the quality of the model, they can be defined as Eq.(3.8). Another aspect need to be evaluated is the computing cost, although there is no specific requirement been addressed yet. It thereby remains one of works for future exploration.

3.6 Conclusion

In this chapter we defined the basic properties of time series. The procedure of time series signal prediction was presented, methods for measuring the prediction were also outlined, moreover, the criteria for model validation were stated.

Subsequently, five efficient prediction approaches, especially the delay coordinate embedding one integrated with AR model, has been introduced. In particular, a data preprocessing method has been stated. Two strategies of achieving multi-step prediction, i.e., with the horizon of prediction greater than one step, has been analyzed.
It was shown that weighted average predictor filters out what it considers to be noise, a feature that may sometimes be quite useful, but that might also be a nuisance, because the user has little control over what the predictor considers noise, and what it considers a signal. Moreover, the choice of threshold value is always ad-hoc. It was also shown that the delay-coordinate-embedding predictor highly depends on the choice of embedding dimension $m$ and number of nearest neighbors $k$ which are both empirical determined oriented.

Finally, the stability of AR model has been discussed. A relevant solution: autocorrelation, to this potential issue in long-term prediction has been mentioned.
4

Implementation

4.1 Introduction

This chapter describes the software design for the time series prediction extension on top of the existing IEEE 1588 protocol stack, PTP daemon. It mainly provides the architecture of the extension built upon regular PTPd, overviews the prediction mechanism, finally, it describes five extension libraries along with their associated subroutines.

The primary objective of this experimental extension is to be able to have various clocks within an IEEE1588 network approximately synchronized in case of loss of GPS or disconnection. The design goals of this extension are real-time prediction, high-resolution (up to 48 hours) drift data simulation, flexible usage of prediction approaches. Especially, it requires to meet the following targets:

1. Near realistic simulation of loss of GPS signal.

2. Real-time prediction. In order to achieve fast forecasting, this extension is required to perform prediction of clock offset efficiently using various tech-
niques and models, e.g., arithmetic average, weighted average, Linear system Model.

3. Efficient data management. The high-resolution clock drift simulation will produce a huge data throughput since it requires retrieve, search, interpolate, and sort operations.

4. Error evaluation. Once the simulation terminates, three types of deviations (Final error $\theta$, i.e., the deviation from the GPS clock by the end of simulation; Maximum error $\Theta$, i.e., the maximum deviation from the GPS clock during the simulation; Actual existing error $\vartheta$ which means the final error without carrying out predictions) will be calculated.

4.2 Overview of PTPd and the Prediction Extension

The PTP daemon (PTPd) is an open source application which implements the Precision Time protocol (PTP) as defined by the version 1.0 of IEEE 1588 standard. This extension of PTPd seeks to carry out a prediction mechanism when the GPS connection fails. It consists of five libraries, and each of them implements a different prediction technique. In order to construct predictors for clock drift estimation, six types of inputs\(^1\) are required:

1. Time disconnected. \textit{(Disconnection\_point)}

2. Time horizon of predictions. \textit{(Disconnection\_duration)}

3. Related previous sequence(s), i.e., a group of sample subsets retrieved from training sets.

\(^{1}\)These parameters are all defined in \textit{interface.h}
4. Threshold value, which serves as a simple filter when comes to weighted average approach. (*Threshold*)

5. Resolution, i.e., the degree to which information about individual sequence elements is preserved. (*Emdims*)

6. Ignored subset, i.e., the length of the initial unstationary part which should be excluded from the training set because of the overhead caused by system initialization. (*Ign_til*)

Moreover, two extra inputs are necessary for Time-series based approaches:

1. Depth, i.e., how far back memory goes. (*Backtrack*)

2. Number of *k* nearest neighbors. (*K_NN*)

Figure(4.1) shows an overview of the major components of the extended PTPd as well as their interactions.

![Figure 4.1: Structure of PTPd with its build-on](image)
The main protocol state machine is implemented as a forever loop with cases for each state. It calls $BMC^2$ after start-up to return the proper state, master or slave, based on the reported clock statistics. After connection has been initialized, msg packer gathers data into or extracts data from the message package as well as the time stamps which were achieved in kernel level. The clock servo computes the offset-from-master from the master-to-slave delay and slave-to-master delays and sends it to the database, once the connection fails, the protocol state machine switches to the prediction configuration and launches the predictor which will retrieve useful history offsets from the database based on its algorithm and calculate the estimation. The estimation will be continuously sent back to clock servo as well as the database. The error counter is implemented as a subroutine of protocol engine associated with the prediction mechanism, it counts the errors based on the estimations and actual offsets stored in the database.

### 4.3 Software Architecture

This section describes the software architecture of this extension. The proposed extension framework will have the following functional components:

1. Module for controlling (start, terminate) the simulation, included as an add-on of protocol engine.

2. Procedures for high-resolution clock drift data prediction, involving data retrieving, constructing predictors, sorting, and data normalization techniques, etc.

3. Module that computes the prediction error.

---

2The best master clock algorithm is used to select the most stable and accurate clock.
The following figure(4.2) and figure(4.3) present the message send and receive paths in a typical system running PTPd, the workflow of prediction within the extended components, and the basic PTPd functions which form the synchronization mechanism. The updated offset data will be continuously sent to data center \((arr, arr\text{Fake})\) from clock servo, while the predictors always take in relevant data from data center and send predictions back afterwards. Error Counter will be called once the simulation is done. This efficient prediction requires the predictors, with the modules, or components of PTPd stack, to work together smoothly and reliably.

![Figure 4.2: Message paths in the PTPd system](image)

The prediction models will continuously take in relevant clock drift data from the data center, subsequently use it to form the predictors, and the offset prediction is carried out efficiently. Meanwhile, the large amount of processing data will be managed in an efficient way to facilitate easy and reliable access. For example, every individual prediction will cause many data retrieves during the procedure.
of model training, the database was implemented using arrays from which the operation of data retrieved will be rapid since the entries of array can be accessed directly. Furthermore, this extension (prediction procedures), its interface to the PTPd stack, and its input modules, will be totally compliant with the PTPd components, as well as the variables passed between PTPd components, while the output data variables and formats will follow PTPd specification. Finally, the original PTPd stack has been slightly modified in order to perform a hard resynchronization at each 2 seconds. Following is the list of functional components:

1. Predictors.
   - Arithmetic average predictor.
   - Weighted average predictor.
   - Directly Extrapolation: delay-coordinate-embedding integrated with average approach.
   - Linear predictor: delay-coordinate-embedding integrated with Auto-Regressive Model.
• Log Transformation: a natural logarithm transformation of the drift data is carried out only if comes to the Log transformation technique which is actually a pre-processing technique.

2. Data Sorting: using the Quick sort Algorithm.

3. Data Retrieve: Euclidean Distance is utilized to determine the $k$ nearest neighbors.

4. Data Storing: two arrays are implemented to store the updated actual offsets and prediction results respectively, and the prediction results will be sent back.

5. Error Calculation: calculate the deviations by comparing the two arrays.

Theoretically, the extension can be configured to run all these models simultaneously.

### 4.4 Data Structures and Components Design

This section describes the internal software data structure. As described earlier, the main components that drive the prediction are the predictors. The sorting and Euclidian-Distance functions are included as subroutines of one of the prediction libraries. Two arrays $arr, arrFake$ passed between PTPd components and the Predictors as well as the error module are used to store the updated real offsets and prediction results respectively, from where the predictors and error counter retrieve the relevant input data. The major components and the subroutines of the extension are:

1. **Data Sorting, quickSort()**

   The data sorting is implemented using an optimized quick sort algorithm
which is different from the Wikipedia version\textsuperscript{3}. Advantages of this optimized one over the Wikipedia code is:

- swap variables: In any one pivot round, the Wikipedia version can pass items through a swap variable many times. The optimized one passes only one item (the pivot) through a swap variable, and only once per round.
- multiple moves: In any one pivot round, the Wikipedia version can move the same item more than once in the list. The optimized one never moves an item to a new position in the list more than once per round.

2. **Data Retrieve, $eud()$**

To identify the nearest neighbors, *Euclidean Distance* has been used for calculating the similarity between two samples.

3. **Solver of Linear equation system, $gauss()$**

*Gauss Elimination*\textsuperscript{4} combined with maximal column pivoting strategy is used for solving the linear equation system which consists of the $k$ nearest neighbors.

4. **Arithmetic average, $Average.c$**

The arithmetic average approach performs the average $aver()$ over the last $n$ offsets stored in the array, i.e., fit the first estimation by averaging over last $n$ observations and iterate to get next estimations by shifting the previous $n$-observations.

5. **weighted average, $Prob.c$**

The weighted average approach is similar to the arithmetic average one, it is

\textsuperscript{3}http://en.wikipedia.org/wiki/Quicksort
\textsuperscript{4}http://en.wikipedia.org/wiki/Gauss_elimination
the same as calculating the weighted average over the previous drift values of which the occurring probability is greater than a given threshold value over the last n rounds. quickSort() function has been called in order to filter useless samples.

6. Delay Coordinate Embedding (direct extrapolation), DusAver.c
   Averaging the results from k nearest neighbors which are found by comparing the Euclidian-Distances eud().

7. Delay Coordinate Embedding (Auto-Regressive), Dus.c
   This is actually called linear regression technique which performs prediction by solving the k linear equations using least-square method. As well as direct extrapolation, it calls function eud() and quickSort() to decide the k nearest neighbors to be used.

8. Log Transformation, Logtrans.c
   A natural logarithm transformation of the stored drift data was carried out, prior to modeling the linear predictor described above.

9. Error, ErrorCount.c
   The final error $\theta$ and error $\vartheta$ (error without prediction) can be calculated by summing the relevant subsets of array arr and arrFake, respectively. i.e., final error equals to the summation of all elements of the array which stores the estimations, and the one stores actual error, respectively. While the possible maximum error $\Theta$ is equal or greater than $\theta$.

10. Protocol Engine add-ons, protocol.c
    Instead of forever loop, in order to terminate the implementation as soon as the GPS signal is back, the protocol.c has been slightly modified. Moreover, two arrays which store the updated drift data have been added as
two extra variables passed between other PTPd functions, e.g., doState(), updateClock(), etc.

11. Clock Servo add-ons, servo.c

The servo has been modified to call a predictor as long as the connection fails, continually updates the arrFake once a prediction has been made.

12. Other code

The interface header file interface.h contains function prototype, structure and data type declarations, as well as constant declarations.

4.5 Conclusion

In this chapter, the scope and purpose of implementation were initially introduced, subsequently, a block diagram provided an overview of the regular system architecture of PTPd associated with the prediction extension which mainly consisted of forecasting libraries and error counter. Two diagrams showed the interaction between the PTPd and the built-ons as well as the message path during the whole extended system. The workflow of how an estimation can be carried out was also demonstrated.

Finally, it described the software data structure, detailed implementations of each components and their subroutines, e.g., sorting, linear equation system solving. Data dictionary, the pseudo code, and user guide are all provided in appendix later.
Experimental Evaluation and Analysis

5.1 Introduction

As already presented before, in this thesis work, accuracy and efficiency are two important criteria for the evaluation of certain prediction techniques. In present chapter, these criteria are evaluated respectively for each of the prediction techniques by applying to the collected drift sequences. Following, their performances for different disconnection scenarios are quantitatively interpreted by showing some plots. Finally, three classes of GPS disconnection are identified based on the test results.

In order to evaluate the accuracy and the efficiency of the arithmetic average, weighted average and time series based prediction techniques, an IEEE1588 network has been set up. The network consists of an independent local area network connecting four different commercial desktops through an Ethernet switch. Each device contains a clock of a different quality but all clocks have a standard deviation in the range of 50µsec/sec. One of the desktop acts as a grand-master clock connected to a GPS receiver. Each node is running Linux and the open source
implementation of IEEE1588 PTPd. The PTPd stack has been slightly modified in order to perform a hard-resynchronization, i.e., call to the `setClock()` system call, at every synchronization round (2 seconds) in order to eliminate the influence of the servo clock tick-rate control. In this work, it requires the final error of prediction to range from $10 \mu\text{sec}$ to $100 \mu\text{sec}$ depending on the considered protection functions of substation automation.

In addition to PTPd, master clock is running an extension library performing the prediction. Each library implements a different prediction technique, i.e. using an arithmetic average approach, a weighted average one and a time series approach. The experiments consist of running PTPd under a constant indoor temperature (around $33^\circ C$ in July) for 24 hours (43200 observations) in a normal configuration, i.e. with the GPS signal connected, and then disconnecting the GPS signal for a delay ranging from 4 seconds to 48 hours. The experiment has been run for each prediction technique on the same PTP slave in order to have comparable results.

The classification of time series has been already outlined in Chapter 2, the training data here were just the copies of history sequences without any synthetically insert, delete or permute. They are natural ones since the original sequences were assumed to be more convictive in reflecting the instantaneous system states if other variables (humidity, temperature, pressure) are not available. Furthermore, during each simulation session, only one single data stream was investigated from a single source observed by a single sensor since multiple recordings were considered as less accurate compared to single recording in terms of linear models [38]. For the validation purpose of different models, after the training data has been stored, the prediction sequences were collected simultaneously in addition to the real drift sequences which can be utilized as validation data set.

A successful prediction of a time series depends on the characteristics of the

\footnote{http://ptpd.sourceforge.net - Version Released on 2007-06-17}
time series. It is to be expected that a stationary process can be predicted better and over a longer time horizon than a non-stationary process. Also, time series that exhibit a more regular, more deterministic behavior should be more easily predictable than time series that exhibit a more stochastic behavior. It is also important to recognize that the time horizon of a meaningful prediction will, at least in this work (ranging from 4 seconds to 48 hours), usually be limited, and in fact, may be rather short.

![History Drift Data](image)

Figure 5.1: Samples of history Drift Data

### 5.2 Overview of Experiments Settings

The objective of experiments is to make one-step ahead prediction using all techniques. There exists different input parameters as well as their values for
different prediction techniques. While the arithmetic average and weighted average predictions are extremely straightforward to implement, the time series prediction needs to be detailed. Before applying these techniques to the data sets of history drifts, a few more general remarks about the experiments setting are in order.

An implementation of a time series prediction has to choose the values for the embedding dimension, \( m \), the number of nearest neighbours, \( k \), and the size of the history data used to make a prediction (depth). From the point of view of predictive accuracy, the performance varies randomly depending on the embedding dimension and number of nearest neighbors, this thesis work considers them as ad hoc. Even though it is impossible to choose the optimal values (i.e. the higher is not necessarily the better), the choice is a tradeoff between the quality of the prediction and the required computing time. In our experiments the history data grows from 43200 elements, i.e., number of synchronization rounds during 24 hours \( \left( \frac{24 \times 60 \times 60}{2} \right) \), to 129599 elements. Since the clock is considered as a stationary process, for the efficiency purpose, only the most recent 200 history subsets or samples are explored from where the training data or nearest neighbors were retrieved. Finally, the choice of value for \( m \) and \( k \) is empirical, from the different experiments, the optimal values seem to be \( m = 8 \) and \( k = 20 \).

One of the evaluation aspects, accuracy, is defined by the final offset \( \theta \) and the maximum possible offset \( \Theta \) during the horizon prediction. The accuracy is therefore defined by:

\[
\theta_m(k) = \sum_{i=1}^{N_t} (\hat{x}_i - x_i) \tag{5.1}
\]

and hence

\[
\Theta_m(k) = \text{Max} \theta_m(k) = \sum_{i=1}^{t} (\hat{x}_i - x_i), \ t \in [1, N_t] \tag{5.2}
\]

\( N_t \) represents the time horizon of prediction, \( \hat{x}_i \) the prediction result while \( x_i \) the real drift.
5.3 Results and Evaluation

The results of the prediction using arithmetic average, weighted average and direct extrapolation can be seen in figure(5.2). It is shown that the three predicted curves are approximately equal to straight lines which almost tend to be narrow bands with the time horizon. The results of the prediction using autoregressive and log transformation can be seen in figure(5.4). It is obvious that the two tricky ones can follow the tracks of actual offsets.

Figure 5.2: The actual and predicted offsets time series using arithmetic average, weighted average and direct extrapolation
Figure 5.3: The actual and predicted offsets time series using AR model and log transformation
• **Final offset - θ**

The first aspect of the evaluation is the comparison of the final offset, i.e., after various horizon prediction ranging from 4 seconds to 48 hours, i.e. 172800 seconds. Figure(5.5) shows the evolution of the θ criteria for the five prediction techniques over a varying horizon prediction. Almost all the five prediction techniques perform well, i.e., with an offset less than 8μsec, for a horizon up to 10 seconds. The weighted average method and autoregressive model both were 5 to 10 times more accurate compared to arithmetic average one which can only stay within 1.5msec offset for a horizon up 5 minutes. It can be noticed that the direct extrapolation has a similar performance as Autoregressive technique, from which the notation that the trend of the predicted time series mainly relies on the number of k nearest neighbors can be inferred. While it comes to longer horizon, e.g. 24 or 48 hours, only the time series prediction integrated with logarithm transformation is able to have an offset less than 500 μsec. Moreover, it is worth noting that even after only 4 seconds of disconnection, the offset is already more than 200 μsec if no prediction is performed.

• **Maximal offset - Θ**

The second aspect of the evaluation deals with the maximal possible offset reached during the horizon prediction. Figure(5.6) shows the evolution of the Θ criteria for the five prediction techniques over a varying horizon prediction. In comparison with the previous evaluation, the autoregressive model based approaches always perform better than the three other techniques. However, when the horizon is less than a couple of dozen of minutes the difference between the weighted average and time series prediction, especially between the two autoregressive based methods, is small and both stay synchronized by less than 10 μsec. When the horizon is in the range of dozen of seconds,
the arithmetic average technique and direct extrapolation method can still give acceptable results since the offset is not bigger than 10 $\mu$sec. Finally, it is interesting to note that in the case of the autoregressive predictor the maximal offset is almost growing linearly with the horizon, while the three other techniques tend to grow exponentially.

- **Processing time**

Finally the last aspect of the evaluation deals with the processing time required to compute the prediction. The processing times reported in figure (5.7) have a similar behavior for each prediction technique since they tend to follow a linear progression. However, the main comment remains that the curves are inverted compared to the two previous figures. The arithmetic average technique is by far the cheapest way to perform a prediction, while the autoregressive based ones are the most expensive one. On an average, the arithmetic average technique requires an average of 0.05 $\mu$sec for each synchronization round, the weighted average and direct extrapolation ones need around 210 $\mu$sec, and the two autoregressive based approaches require 315 $\mu$sec and 350 $\mu$sec respectively.

### 5.4 Comparison of Fast Forecasting Methods

From the results presented in the previous section, three classes of GPS disconnection are identified:

1. In the case of a short disconnection, i.e., in the range of a couple of seconds due to for example atmospheric disturbances, a arithmetic average prediction allows to stay synchronized by less than 10 $\mu$sec with a negligible processing overhead. Moreover, with such a low processing overhead each node of a
Figure 5.4: Evolution of the $\theta$ criteria for the five techniques over a varying horizon prediction.
Figure 5.5: Evolution of the $\Theta$ criteria for the five techniques over a varying horizon prediction
Figure 5.6: Evolution of Processing time for the five techniques over a varying horizon prediction.
substation can run its own arithmetic average prediction. This scenario will allow each node to handle the loss of one or two IEEE 1588 synchronization messages in a row due to networking issues.

2. In the case of a long disconnection, i.e., from one hour to two days due to for example the loss of the GPS antenna, a time series prediction is the only possible way to keep the substation synchronized. Even though after two days of prediction the offset is in the range of $75 \mu sec$ some protection functions will have to be turned off, their resynchronization can be done smoothly (i.e., instead of a full reinitialization) and protection functions with looser synchronization requirements can keep running. However since this kind of prediction is time consuming, it has to be implemented on the grand master node of the substation.

3. In the case of a medium disconnection, i.e., couple of minutes due to for example small maintenance operations such network cable replacement, a weighted average prediction is the best tradeoff in terms of offset and processing power. However since this kind of prediction can not take place on every node of a substation because of its processing requirements, some specific nodes have to be identified. Such scenario leads to the notion of synchronization group identified [39] in where the prediction will run on each group leader node.

### 5.5 Conclusion

This chapter firstly described the experiments settings as well as the properties of the offset time series which were used for training and validation of later simulation. Following, it reported the prediction results obtained with five different methods for the offset time series, three criterions based on final prediction error,
possible maximal error and processing cost were briefly evaluated and commented. Finally, the discussions were narrowed down to potential applications to the substation automation. Three classes of disconnection scenarios, i.e., short, medium, long term disconnections were identified from the result of evaluation.
Conclusion and Future Work

6.1 Conclusion

This master thesis project presented an overview of the work carried out for the project titled *Improving Reliability of IEEE 1588 Protocol in Electric Substation Automation based on Clock Drift Prediction*. It relates to the field of time synchronization in substation automation (SA) using the IEEE 1588 protocol, while the aim of this project is to handle the un-addressed loss-of-GPS issues (which are common scenarios in case of an electric substation) by keeping the substation synchronized to the GPS time or at least as close as possible, of which the substation can manage to avoid a hard resynchronization of the clock’s devices and therefore a full re-initialization of the protection functions when the GPS signal is back.

In terms of the reliability of electric substation automations, one pre-requisite to perform efficient protection functions is to have synchronized data provided by the various devices, and the synchronization requirement ranges from 10 μsec to 100 μsec depending on the considered protection functions. Another naturally arisen problem is the extra processing cost of carrying out those predictions. In
First, the various time prediction approaches of precision clock were introduced. These methods were classified into three categories: *physical clock modeling*, *time series prediction* and *direct extrapolation*. The currently used prediction methods for precision clocks were investigated, subsequently, the discussion of how this thesis been confined to the last two categories of approaches was also presented. Following, the procedure for time series prediction was outlined. The steps identified in this procedure were: collection of data, formation of a set of candidate models, selection of criterion of model fitness, model identification and finally model validation. The two simple approaches such as *arithmetic average* and *weighted average* were presented. Then, a detailed description of the *delay coordinate embedding* (DCE) algorithm was provided. Finally, methods for data pre-processing as well as measuring the prediction accuracy (validation) were also presented.

Besides the two simpler arithmetic average and weighted average methods (direct extrapolation), the more tricky ones which are based on linear autoregressive (AR) model were implemented for this thesis project. In order to simulate the application scenarios of IEEE 1588 network, the implementation part of this thesis work carried out five different prediction techniques which were built upon the open source implementation of IEEE 1588 PTPd.

The implemented methods were tested on the clock offsets time series collected from IEEE 1588 networks. The results of those methods presented in section 5.3 were analyzed and compared with each other, subsequently, three classes of GPS disconnection were identified. In general, the arithmetic average method can have negligible processing overhead and the autoregressive based methods can have the best prediction accuracy. Although the results of arithmetic average one obtained with short-term disconnection was good, its performance showed that it has not
the ability to satisfy the synchronization requirements while comes to long-term disconnection. Therefore, the arithmetic average method can be considered as one potential approach to handle the short-term GPS disconnection. On the contrary, the autoregressive based algorithm showed that it has the ability to provide a satisfying accuracy over a period of 48 hours (long-term GPS disconnection) but at a much higher computing power cost. In case of medium disconnection, the weighted average method is the best tradeoff in terms of accuracy and processing power.

The ability of the AR model based algorithm (DCE) was best seen with integration of a logarithm transformation in the initial phase of prediction. Another feature of the DCE algorithm is that its error exhibited linear increment with the time horizon while the others were exponential. In conclusion, it can be said that the delay coordinate embedding algorithm gave very encouraging results and has the potential for further improvements.

The results show that the DCE algorithm integrated with a logarithm transformation can provide an accuracy of less than 80 microseconds over 48 hours of prediction but at a high computing cost, while using a arithmetic average based prediction technique an accuracy of less than 10 microseconds over a couple of seconds can be achieved at an extremely low cost.

Those results demonstrate that clock drift prediction can be used in the context of electrical substation automation in order to improve the overall protocol reliability in SA without modifying the protocol itself. Moreover, compared to the redundancy concept for the master node, the same level of reliability is achieved at no additional hardware cost.
6.2 Future Work

The results obtained with the implemented methods are very encouraging. In all problems tested there was a huge decrease in the final deviation compared to the actual deviation without any prediction. Although the results and analysis are promising and show evident benefits, a couple of practical issues remain in order to deploy the technology in a substation.

- In the case of a delay coordinate embedding prediction, the main problem as the choices of the embedding dimension and number of nearest neighbors identified by [38] still remain. The quality of the DCE prediction depends highly on the \( m \) and \( k \) parameters. However, there is no mean to evaluate the optimal values for those parameters and only an empirical solution can be foreseen. Since \( m \) and \( k \) depend on the drift produced by each clock, a set of experiments have to be run on every node which is susceptible to perform such prediction.

- If the synchronization rounds are too short, the weighted average or time series prediction combined to the other tasks running on a node may not have enough time to proceed.

- The experiments presented in this thesis project assume that a node knows immediately if the GPS signal or master connection is lost. However in a real case, immediate notification of a disconnection remains an issue. As an example, without modifying the IEEE 1588 protocol, at least one synchronization round is required to detect a disconnection and therefore allows the clock to drift during this time without correction. Even though one solution is to shorten the synchronization round, one must be aware of the computing overhead introduced by the prediction especially in the case of the time series
In conclusion, the results presented in the thesis are promising, however, they need to be consolidated by running the prediction techniques on various different clock quality and different environment conditions. Moreover, an FPGA based implementation of the different prediction techniques needs to be implemented in order to off-load the main processor and therefore smoothly switch-over the different prediction techniques as the prediction window grows. Reliability for synchronization protocols has mainly been tackled from an architectural point of view [39] [40], e.g., relying on redundant time servers and multiple network paths, it can easily be combined with the idea proposed in this thesis work.
A.1 Data Dictionary and Pseudo Code

1. Average.c

- Function Name - `average`
- Parameters
  - (a) `*arrFake`: Pointer to array
  - (b) `arrLen`: int
- pseudo code

```plaintext
/* average(arrFake, length) */

average(arrFake, length)
{
    while i<=length
    do
        sum(arrFake(i)); // sum up previous observations;
    end;
```
2. Prob.c

- Function Name - `proba`
- Parameters
  - (a) `*arrFake`: Pointer to array
  - (b) `arrlen` : int
- pseudo code

```
/* proba(arrFake, length) */
proba(arrFake, length)
{
    while i<=length
    do
        if probability( arrFake(i) ) > THRESHOLD
        {
            sum=sum + arrFake(i)* times(arrFake(i))/length;
            prob=prob + times(arrFake(i));
        }
    end;
    return sum/prob ;
}
```

3. dvsAver.c

- Function Name - `dvsAver`
- Parameters
  - (a) `*arrFake`: Pointer to array
(b) arrLen: int

- pseudo code

\[
\begin{align*}
&\text{dvsAver(arrFake, length)} \\
&\{ \\
&\quad \text{while } i\leq \text{length} \\
&\quad \quad \text{do} \\
&\quad \quad \quad \text{subset}[i]=\text{locate(arrFake)}; \\
&\quad \quad \quad /* \text{Euclidian Distance} */ \\
&\quad \quad \quad \text{eud(testVector,subset}[i], \text{length}); \\
&\quad \quad \text{end;} \\
&\quad \text{while } i\leq K \\
&\quad \quad \text{do} \\
&\quad \quad \quad /* \text{sum up the K elements succeeding the K instances found} */ \\
&\quad \quad \quad \text{sum(subset}[m]); \\
&\quad \quad \text{end;} \\
&\quad \text{return sum}/K; \\
&\} \\
\end{align*}
\]

4. DVS.c

- Function Name - \textit{dvs}

- Parameters

(a) *arrFake: Pointer to array

(b) arrLen: int
pseudo code

/* dvs(arrFake, length) */
dvs(arrFake, length)
{
    testVector = locate(arrFake);  // locate the lastest subsets
    while i<=length
        do
            subset[i]= locate(arrFake);
            eud(testVector, subset[i], length);
        end;

    /* form the K linear equations */
    while i<=k
        do
            matrix[i]= subset;
        end;

    /* Gauss elimination Algorithm */
    coefficient = gauss(matrix);
    /* Cartesian product */
    return Cartesian(coefficient, testVector);
}

5. logtrans.c

• Function Name: logtrans

• Parameters:

  (a) *arrFake: Pointer to array
(b) arrLen: int

• pseudo code

```c
/* logtrans(arrFake, length) */
logtrans(arrFake, length)
{
    testVector = locate(arrFake); // locate the testing subset
    while i <= length
        do
            subset[i] = locate(arrFake);
            eud(testVector, subset[i], length);
        end;

    /* Logarithm Transformation */
    while i <= k
        do
            matrix[i] = log(subset);
        end;

    /* using Gauss Elimination - maximal column pivoting */
    coefficient = gauss(matrix);

    /* Cartesian product */
    return Cartesian(coefficient, testVector);
```

6. Error_Count.c

• Function Name: `errorCount`
• Parameters:
  (a) *arr: Pointer to array
  (b) *arrFake: Pointer to array

• pseudo code

    /* errorCount(arr,arrFake) */
    errorCount(arr,arrFake)
    {
        for (i=DisconnectionTime; i<=DisconnectionDuration; i++)
        {
            originalDev =sum(arr[i]);
            finalError= sum(arr[i]-arrFake[i]);
        }

        /* maximum positive value or minimum negative value */
        maxError= max(finalError);
    }

A.2 Guideline for Users

This section is useful for developers who want to add and/or modify procedures in this PTPd extension, as well as playing around with it.

The default configuration of this extended PTPd is entirely compliant with the regular one, there exists a specific source code documentation of PTPd on the website\(^1\). For the prediction part, the libraries are all allocated in the same src directory as well as other PTPd source files, while the headfiles have been all

\(^1\)http://ptpd.sourceforge.net/doc.html
wrapped under the *include* directory.

To run and test different prediction techniques, one can customize the input parameters (e.g., time disconnected, time re-connected) to control the simulation by accessing to the *interface.h* from where all the constants were defined. To use different predictors, one can modify the clock servo (*servo.c*) by changing the library it calls, for instance:

```c
void updateClock(*rtOpts, *ptpClock, *arr, *arrFake,*counter)

/* GPS signal is available */
if(*counter <=Disconnection_point)
{
    ...
};

/* loss of GPS signal */
else
{
    ...
    prediction = dvs(arrFake,*counter);  \ calls dvs predictor
    arrFake[(*counter)-1] = prediction;  \ update the training data
    ...
}
```


Bibliography


