Degree Project in Machine Learning
Second cycle, 30 credits

Time synchronization error
detection in a radio access network

MOULIKA MADANA
Time synchronization error detection in a radio access network

MOULIKA MADANA
Abstract

Time synchronization is a process of ensuring all the time difference between the clocks of network components (like base stations, boundary clocks, grandmasters, etc.) in the mobile network is zero or negligible. It is one of the important factors responsible for ensuring effective communication between two user-equipments in a mobile network. Nevertheless, the presence of asymmetries can lead to faults, making the detection of these errors indispensable, especially in technologies demanding ultra-low latency, such as 5G technology. Developing methods to ensure time-synchronized mobile networks, would not only improve the network performance, and contribute towards cost-effective telecommunication infrastructure. A rule-based simulator to simulate the mobile network was built, using the rules provided by the domain experts, in order to generate more data for further studies. The possibility of using Reinforcement Learning to perform fault detection in the mobile network was explored. In addition to the simulator dataset, an unlabelled customer dataset, which consists of time error differences between the base stations, and additional features for each of its network components was provided. Classification algorithms to label the customer dataset were designed, and a comparative analysis of each of them has been presented. Mathematical algorithm and Graph Neural Network models were built to detect error, for both the simulator and customer dataset, for the faulty node detection task. The approach of using a Mathematical algorithm and Graph Neural Network architectures provided an accuracy of 95% for potential fault node detection. The feature importance of the additional features of the network components was analyzed using the best Graph Neural Network model which was used to train for the node classification task (to classify the base stations as faulty and non-faulty). Additionally, an attempt was made to predict the individual time error value for each of the links using Graph Neural Network, however, it failed potentially due to the presence of fewer features to train from.

Keywords

GNSS - Global Navigation Satellite System, OAS - Over-the-air-synchronization, PRTC - primary reference time clock, PTP - precision time protocol, Gauss Jordan elimination, GNN- Graph Neural Network
Sammanfattning


Nyckelord

GNSS - Globalt navigationssatellitsystem, OAS - Över-the-air tidssynkronisering, PRTC - Primär referenstidklocka, PTP - Precisionstidprotokoll, Gauss Jordan eliminering, GNN- Graf neurala nätverk
Sammanfattning
Acknowledgments

First of all, I would like to extend my deep and sincere gratitude to Ann-Helen Kalander, Alexandra Mikalesson, and Mikael Johansson, for providing me an opportunity to work as an innovation intern at Ericsson and for providing my master’s thesis work. I express my gratitude to Tahar Zanouda, with whom I had discussions, and they played an important role in helping me to effectively complete this thesis work. I would also cherish the memories I created with wonderful people at the RAN 5G Synchronization department at Ericsson, during fikas and the group outing. I am thankful to György Dán and Kiarash Kazari, my examiner and supervisor at KTH, for guiding me with my documentation work.

However rocky things go, there are two people in my life, who have proved over time that, they would love me for whoever I am, and keep supporting me through thick and thin: my parents. I am deeply grateful to them. I am also fortunate to have a younger sibling who taught me a lot just being himself. I express my heartfelt gratitude to all my friends from KTH and Amrita.

I would also like to acknowledge the assistance provided by ChatGPT, Grammarly, and Google Translate for paraphrasing, correcting the grammar of my sentences, and translating the abstract to Swedish, for this thesis report.

Stockholm, October 2023
Moulika Madana
Contents

1 Introduction ........................................ 1
   1.1 Background ........................................ 1
   1.2 Problem ........................................... 4
       1.2.1 Problem statement and definition .............. 4
       1.2.2 Research questions addressed .................. 4
   1.3 Purpose ........................................... 4
   1.4 Sustainability and ethical aspects ................... 5
   1.5 Research methodology ................................ 5
   1.6 Delimitations ....................................... 6
       1.6.1 Challenges ..................................... 6
       1.6.2 Delimitations ................................... 7
   1.7 Structure of the thesis ............................... 7

2 Background ........................................... 9
   2.1 Mobile network elements ............................. 9
   2.2 Time synchronization ................................ 10
       2.2.1 Need for time synchronization .................. 10
       2.2.2 How time synchronization is achieved? .......... 11
           2.2.2.1 RAN-based time synchronization .......... 12
           2.2.2.2 Network-based time synchronization ....... 12
       2.3 Mobile network as a graph ........................ 14
   2.4 Reinforcement learning ................................ 14
   2.5 Graph neural network ................................ 17
       2.5.1 Working of GNN ................................ 19
       2.5.2 GCN ........................................... 19
       2.5.3 GraphSAGE ..................................... 20
       2.5.4 HGNN ......................................... 20
   2.6 F1 score ............................................ 20
   2.7 Dealing with imbalanced dataset ...................... 21
2.7.1 SMOTE .................................................. 21
2.7.2 SMOTE-N ............................................. 22
2.7.3 SMOTE-NC ........................................... 22
2.8 Related works ........................................... 22
2.8.1 GNN for classification and regression .......... 23

3 Methods ...................................................... 25
3.1 Mobile network data structure ....................... 26
3.2 Initial hypothesis ....................................... 27
3.3 Simulator development ................................ 27
3.4 Failure of the hypothesis .............................. 29
3.5 Mathematical techniques for fault detection .......... 31
3.6 Labeling customer dataset ............................ 32
3.7 GNN models ............................................. 34
3.8 Software tools and frameworks ..................... 35

4 Results and Discussion .................................. 37
4.1 Data Collection and Data Preparation ............... 37
4.2 Qualitative results ..................................... 38
  4.2.1 How accurate were the labeling techniques used? 38
    4.2.1.1 Simulator dataset and heatmaps ... 38
    4.2.1.2 Conclusions from the study .......... 38
  4.2.2 GNNs for regression ................................. 40
4.3 Quantitative results .................................... 40
  4.3.1 Mathematical methods results .................. 40
    4.3.1.1 Simulator dataset ....................... 40
    4.3.1.2 Results for simulator dataset ....... 41
    4.3.1.3 Results for the customer dataset ... 41
  4.3.2 GNN architectures ................................ 41
  4.3.3 GNN experiments and results .................. 41
  4.4 Discussion about the research questions .......... 43

5 Conclusions and Future work .......................... 45
5.1 Conclusion ............................................. 45
5.2 Limitations ............................................ 46
5.3 Future work ........................................... 46

A Mathematical methods for solving equations ...... 47
  A.1 Row echelon form and Reduced row echelon form 47
  A.2 Gauss Jordan elimination ........................... 48
List of Figures

1.1 Mobile Network, referred from [3] ................................. 3
2.1 Time synchronization in TDD ............................ 11
2.2 Simplified mobile network ........................................ 14
2.3 The RL method .................................................... 16
2.4 Sample of the Heterogenous graph generated for the customer dataset ........................................... 18
2.5 Sample of the Homogeneous graph generated for the customer dataset ........................................... 18
3.1 Time error difference calculation in a mobile network ................................. 27
3.2 The screenshot of the web page interface for the simulator dataset ........................................... 28
3.3 Simulator dataset versus the Customer dataset ........................................... 29
3.4 Issues using RL .................................................... 30
3.5 Mathematical method for fault detection ........................................... 32
3.6 Classification Algorithm 1 ........................................... 33
3.7 Classification Algorithm 2 ........................................... 34
4.1 Heatmaps for the Classification Algorithms ........................................... 39
4.2 Feature importance analysis ........................................... 42
List of Tables

4.1 F1 score (mentioned in Section 2.6) for the model in order to check for feature importance . . . . . . . . . . . . . . . . . . . . . . . . 43
xiv | List of Tables
Chapter 1

Introduction

1.1 Background

Ericsson is a multinational telecommunication and networking company focusing on building top-notch technologies, and revolutionizing the way we communicate, work, and live, by enabling seamless communication and connectivity across the globe [1]. To achieve this goal, establishing effective and efficient communication in a mobile network is vital. The effectiveness of communication in a mobile network is governed by a number of parameters, one of which is "Network Synchronization". When two user equipments (e.g., mobile phones, IoT devices) communicate with one another, the data from one device is split into smaller units called data packets and sent across various networking components in the mobile network to reach the other device. To ensure efficient communication, without data loss, all the clocks present in each of the networking devices are expected to operate at the same time and frequency. Network synchronization [2] refers to the process of ensuring all the clocks present in each of the networking devices are operating at same time and frequency, ensuring that data packets are transmitted and received seamlessly between user equipments, guaranteeing reliable communication. One of the integral components of Network synchronization is, 'Time synchronization'. 'Time synchronization' refers to the process of maintaining accurate and consistent time among the different network elements, such as base stations, switches, and other network devices. There are several issues that are solved via Time Synchronization. Reducing collision and interference between different signals, hence preventing data loss, ensuring efficient coordination between different cells, hence providing consistent service to mobile users,
and providing ultra-low latency and high reliability for 5G technology \[3\] \[4\] are some of the many benefits of it. The importance of time synchronization is further elaborated in Section 2.2.

A mobile network, shown in Fig.1.1, consists of various network components, like grandmasters, routers, base stations, and user equipments. In Fig.1.1, the components of a mobile network are divided into three domains. They are T-GM(Telecom- Grandmaster), Transport domain, and RAN domain. T-GM consists of PRTC(Primary Reference Time clock), which are expensive devices receiving accurate time from the GNSS(or the satellite). In the transport layer, we have routers and switches, which may or may not contain the boundary clocks (which are responsible for keeping track of the time). In the RAN domain, we have Base stations as DU(Digital Unit), which consists of cells, which are also referred to as RU(Radio Unit). RUs are the first networking components the user equipment would contact while establishing communication between two user equipments. Each of the components has a clock to keep track of the time \[5\]. More accurate time is received by grandmasters, the first networking devices present in the mobile network which receive the synchronization message \[6\].

To establish a successful time-synchronized communication between two user equipments(like mobile phones or IoT devices), it is crucial that the clocks present in the base stations connected to those user equipments should be operating with no time difference or time error.

The 'time difference' is the difference in the time of the clocks present in the two networking components. The time difference would be considered as a 'time error difference' or 'time error', in case, the time difference exceeds a certain threshold. For the purposes of this thesis work, domain experts recommended a suggested time error threshold of 1500ns among the base stations. However, it is also important to note that, to establish time synchronization, the ideal time difference should be 0ns, and anything greater than that is a time error. Hence, time difference, time error difference, and time error can be used interchangeably \[3\].

During this study, in order to measure the time difference at the base stations, OAS(Over the Air Synchronization) is used at Ericsson \[3\]. In order to ensure time synchronization, PTP(Precision Time Protocol) is employed \[7\]. More details about the PTP protocol, OAS solution, and different networking components are present in Chapter 2.

The fundamental idea of the PTP protocol is to use PTP messages from the PTP grandmaster to the leaf PTP clocks. The protocol is built on the exchange of timing messages in both directions. This message exchange makes it possible
to estimate the route delay and distribute the time from the PTP grandmaster [8].

The OAS solution [3] is a unique solution to find time alignment or time difference in the RAN domain at Ericsson, which uses time information between neighboring base stations and compares them to return the time error differences value, as shown in Fig. 1.1, which will aid in the detection of faulty connections later.

It is important to note that both PTP and OAS assume symmetrical time error, which means, for two networking devices, the time difference is assumed to be the same in both directions [3].

Despite employing these protocols, various factors such as failure of the symmetrical time error assumption, (due to differences in the cable types used in both directions for two network components, or, differences in cable length or faulty cable, etc.), presence of faulty networking component, etc, contribute towards a fault in time synchronization, and time error is still observed. This thesis work is an attempt to explore and implement methods to detect such faulty equipments via machine learning algorithms.

Figure 1.1: Mobile Network, referred from [3]
1.2 Problem

1.2.1 Problem statement and definition

Achieving time synchronization is crucial for not only the telecommunication industry but also in other domains like financial services [9], energy and consumption [10], healthcare [11] etc. However, achieving time synchronization is challenging because of various reasons like clock drift (where clocks of the two network devices do not operate at the exact speed as another clock) [12], presence of asymmetries [3] etc.

At Ericsson, the time error differences are measured between the networking components in RAN domain, as shown in Fig. 1.1, like the base stations using OAS solution. However, pinpointing which specific network components, present in the T-GM domain or the transport domain are responsible for causing these time errors remains a challenge.

The components using the PTP protocol, where there is an asymmetry in the packet paths could be the source of the time errors. Asymmetries refer to differences or imbalances in the time it takes for synchronization messages to travel between devices or the variability in these message transmission times.

The focus is to detect the components of the mobile network that may be introducing asymmetries or faults contributing to the hindrance in time synchronization in mobile networks. Detecting these components and adding a correction would help in reducing the time error difference between the base stations, obtained via OAS, and aid in time synchronization.

1.2.2 Research questions addressed

Based on the problem statement mentioned above, this thesis aims to investigate the following research questions:

- What are the various techniques one could use to identify the faulty nodes or in other words, the most suitable nodes where one could apply asymmetry compensation?
- How accurate are those methods?

1.3 Purpose

Developing effective methods for troubleshooting and issue resolution is crucial for ensuring reliable time synchronization in the telecommunications
industry. Current approaches rely on a combination of technical expertise and diagnostic tools, with one example being the use of the PTP protocol to incorporate a time error compensation value. The goal is to design and implement methods to pinpoint the faulty elements to enhance time synchronization. Once identified, these components can either be replaced or adjusted by adding an asymmetry compensation value. This approach aids in detecting faulty equipment, thereby reducing the overall cost of identifying the root cause of synchronization issues. By implementing such methods, we leverage the advantages of precise time synchronization, which include improved network performance, reduced latency, and enhanced reliability. These efforts contribute to a more efficient and cost-effective telecommunications infrastructure, ensuring that critical applications and services benefit from accurate time synchronization.

1.4 Sustainability and ethical aspects

Integrating fault detection techniques helps reduce labor expenses related to addressing time synchronization issues. Furthermore, the project’s ethical and sustainability aspects are reinforced by the benefits of accurate time synchronization. This includes ensuring minimal latency for 5G technologies, which improves the efficiency and responsiveness of telecommunications networks. Additionally, reducing data loss and collisions not only enhances network reliability but also promotes efficient resource utilization, in line with environmentally conscious and ethical principles in technology infrastructure development.

1.5 Research methodology

The initial hypothesis to deal with the presented problem statement was to train a Reinforcement Learning (RL) agent to predict the time errors at all the nodes and links, and based on the time error difference present at the base stations, predict the faulty nodes. However, a substantial amount of data is needed to train an RL model. Consequently, a simulator was built, based on the rules provided by the domain experts. However, due to reasons mentioned in Chapter 3, the hypothesis to use the RL model was not an optimal solution, and alternative methods were explored. Mathematical methods like Gauss Jordan elimination, which is described in detail in Chapter 2 were employed in order to detect the fault, by converting
the mobile network to a set of equations. However, while this approach successfully pinpointed defective cells, it occasionally included non-faulty ones as well. Nonetheless, it was observed that the set of genuinely faulty cells typically formed a subset of the identified faulty cells in most of the cases. The results of this analysis are described further in Chapters 3 and 4. To get further closer to the faulty set of nodes, other methods were explored.

An unlabelled customer dataset, with two additional features for the base stations, was provided. Homogenous and heterogeneous mobile network graph data were derived from this dataset. The goal was to assess the importance of the two additional features. However since the dataset was unlabelled, the classification of base stations was performed using two distinct classification algorithms, mentioned in Chapter 3. After assessing which algorithm is suited better for the real world, one algorithm was chosen, which is presented in Chapter 4. Subsequently, Graph Neural Networks were used on both the simulator and customer dataset to check their effectiveness in detecting the faulty nodes and, analyzing the feature importance for the customer dataset. The results of this study are presented in Chapter 4.

1.6 Delimitations

Some of the delimitations and challenges encountered during the thesis work are mentioned below.

1.6.1 Challenges

- The customer dataset, which would be further described in later sections, was not a complete graph, but a partial graph, and the components in between them were unknown.

- The presence of fewer features to learn from. There were only two additional features to learn from and only for the base stations in the customer data.

- The customer dataset was unlabelled, posing a challenge during node classification. Hence, classification algorithms were formulated as mentioned earlier.

- The dataset was imbalanced, that is the number of faulty links was lower than the number of non-faulty links in the customer dataset.
1.6.2 Delimitations

- It was assumed that the time synchronization technique used in the mobile network is the PTP protocol.

- The time error difference between two nodes in the base station was assumed to be symmetrical.

- The simulator was generated based on rules suggested by the domain experts, but it could be further improved to take an algorithm like Best Master Clock Algorithm, to choose the path.

- The designed classification algorithms don’t work in all the possible cases, as elaborated further in Chapter 4.

1.7 Structure of the thesis

- Chapter 1 provides a comprehensive overview, delving into the background, problem statement, and study’s objectives.

- Chapter 2 provides an elaborate overview of the elements comprising the mobile network, the data structure within it, as well as the reinforcement learning, machine learning, and mathematical algorithms utilized in this thesis. The chapter delves into a detailed description of these components and their relevance to the research conducted in this work.

- Chapter 3 presents a sequential account of the methods and research conducted throughout the thesis, along with the underlying motivations behind the study. This section addresses questions related to the procedures employed, the various studies carried out, and the rationale behind conducting the research.

- Chapter 4 involves a discussion of the outcomes obtained from the studies conducted, as mentioned in Chapter 3. The chapter critically analyzes and interprets the results achieved during the research, providing an in-depth exploration of their implications and significance within the context of the thesis work.

- Chapter 5 encompasses the conclusions drawn from the research, considerations for future work, and reflective insights.
8 | Introduction
Chapter 2

Background

2.1 Mobile network elements

"A cellular network or mobile network is a telecommunications network where the link to and from end nodes is wireless and the network is distributed over land areas called cells, each served by at least one fixed-location transceiver (typically three cell sites or base transceiver stations)" [5].

The mobile network has evolved from 1G to 5G [13] since 1979, when the first commercial cellular network(1G) was launched in Tokyo, by Nippon Telegraph and Telephone (NTT). The ‘G’ in the mobile network type stands for generation. The launch of each generation of mobile networks unlocked new applications. 1G was used to establish basic analog voice communication, examples included AMPS (Advanced Mobile Phone System) in the United States and the NMT (Nordic Mobile Telephone) System in Europe. 2G introduced digital voice encoding, which improved voice quality and reduced interference, examples include, D-AMPS (digital AMPS technology) and cdmaOne (Code Division Multiple Access) in the USA, through to PDC (Personal Digital Cellular) in Japan. 3G marked the beginning of mobile internet access. It provided faster data transfer speeds, enabling applications such as web browsing, email, and video calls on mobile devices. 4G significantly improved data speeds and reduced latency. 5G offers massive IoT connectivity, greater rates of data transfer, lower latency, and improved reliability in comparison to the previous generation of the mobile network.

To establish communication between two user equipments (like mobile devices or IoT devices etc.), a series of networking devices present in the mobile network communicate with one another to transfer data. These components are, user equipment (UE) or mobile devices, Base stations or cell towers, Radio
Access Networks (RAN) [14], etc. The user equipment is the end-user devices, like smartphones, tablets, or any wireless-enabled devices, that communicate on the mobile network. The base station is the fixed transceiver that is responsible for establishing communication between one or more wireless mobile devices and can provide coverage in a specified geographic location, which is a part of RAN domain. RAN consists of all the networking devices such as radio equipment, and antennas, that ensure the communication is happening between the UE and the base station.

Some of the key components to establishing seamless communication in mobile networks include network synchronization, power management, latency reduction techniques (like Semi Persistent Scheduling (SPS) and short Transmission Time Interval (sTTI) [15]), etc. Time synchronization in mobile networks is crucial, as it plays a key role in distributed data acquisition, and, real-time communication.

### 2.2 Time synchronization

#### 2.2.1 Need for time synchronization

User equipments communicates with the base stations either using TDD(Time Division Duplex) or FDD(Frequency Division Duplex) [16]. FDD and TDD are two distinct duplexing techniques used in wireless communication systems. FDD allocates separate frequency bands for transmitting and receiving data simultaneously, allowing for continuous two-way communication [16]. In contrast, TDD divides the available spectrum into time slots, alternating between transmitting and receiving within the same frequency band [16]. Time synchronization plays a crucial role in TDD [3].

The scheduling of the time slots happens via, downlink(DL) and uplink(UL) time scheduling. During UL, only the user equipments communicate with the base stations, and during DL, only the base stations communicate with the user equipments. The number of UL and DL slots is determined by the amount of traffic in the mobile network. For example, in social media applications, video conferencing, or user-generated content platforms, users often upload data (UL) more frequently than they download content (DL). And when users post videos or images on social media, the UL traffic can be substantial. When users stream content, such as videos or music, from servers or content providers, the DL traffic dominates. Now all the clocks among the base stations need to be synchronized in order to ensure no data loss or collision caused by overlap of
UL and DL [16] [3]. However, to accommodate for a time difference, guard time is added, during the scheduling when none of the devices communicate with one another [3]. An example of the scheduling is shown in Fig. 2.1. This image has been taken from [3].

In this figure, to establish communication between two user equipment $A$ and $B$, the user equipment communicates via Base stations $BS_A$ and $BS_B$ respectively. However in the figure, due to a minute difference in the clocks for the two base stations, the DL for $BS_B$, continues further in the guard time. However, due to the presence of guard time, there is no collision or data loss in the figure. However, one can imagine a scenario where there would be an overlap when the time difference between the base stations is larger than the guard time. Hence to avoid such cases base stations need to be time synchronized [3].

2.2.2 How time synchronization is achieved?

The synchronization solutions for evolving mobile networks have significantly improved, as we transitioned from 3G to 5G. During the era of 3G, Synchronous Ethernet (SyncE) technology was commonly employed to synchronize 3G base stations. A centralized source (Cesium clocks), provided a uniform frequency reference to 3G base stations by transmitting
SyncE signals through Ethernet equipment [17]. This process ensured synchronization of the base stations to maintain precise timing and network reliability.

For 4G and 5G mobile networks, more precise synchronization techniques are important.

In order to achieve time synchronization, mobile networks rely on various time synchronization methods, e.g., PTP protocol and local time reference provided by GNSS [18] receiver that receives time from satellite systems.

2.2.2.1 RAN-based time synchronization

RAN-based synchronization solutions are the solutions that achieve synchronization from the devices present in the RAN domain [3]. One such example is GNSS-based synchronization at Base stations. In this solution, compact and low-priced devices equipped with a GNSS receiver and a reliable internal clock can be installed specifically in the devices in the RAN domain, e.g., Base stations, where they are needed without additional intelligence in the network itself [19]. This technique is used in TDD macrocells to provide phase/time with the proper accuracy [20]. However, the disadvantage of this method is that the GNSS receiver requires sky visibility, and may experience outages (intentional or unintentional). It is therefore usually not applicable for indoor small cells scenarios as well as for some particular outdoor small cells scenarios [20].

2.2.2.2 Network-based time synchronization

When synchronization is achieved through the transport network, it falls under the category of a "Transport-based solution." Protocols like PTP and NTP are an example of this kind of solution [3].

Precision Time Protocol (PTP) is a time synchronization protocol, which is specifically designed for applications requiring highly accurate timing, to nanosecond precision. PTP utilizes a master-slave architecture with a Grandmaster clock as a master clock or a primary time reference clock which distributes a time reference to all the other devices or slave devices. The Best Master Clock Algorithm (BMCA) [21] is used to determine the best grandmaster in a network.

PTP is defined by IEEE 1588 standard [22] and is used in industries that rely on highly accurate and synchronized time, such as telecommunications, financial trading, power utilities, etc. Key components of this protocol include
grandmaster clocks, slave clocks, boundary clocks, etc. Grandmaster refers to a reliable and highly accurate time reference that serves as the primary reference clock for synchronizing all clocks and devices within a network. It acts as a source of time for the entire network. Devices in the network that synchronize their internal clocks to match the time provided by the grandmaster clock are Slave clocks. Boundary clocks are intermediate devices that act as a bridge between different PTP devices. There are several intermediate boundary clocks in a mobile network. All the PTP clock devices except the grandmaster have a Parent port clock. It is important to note that the boundary clocks and grandmasters are a part of LAN (Local Area Network) or WAN (Wide Area Network) [23], and they are present in the routers and switches, which are devices used in the core network of a telecommunication system and are responsible for routing data packets between different networks. It is important to note that PTP protocol assumes symmetry in time error, which is that the delay is equal in the two transmission directions as highlighted in Fig. 1.1 using orange arrows.

PTP was created to address the limitations of other time synchronization methods like Network Time Protocol (NTP) [24], to meet the stringent timing requirement for industries like telecommunications. Using PTP protocol the clocks determine an offset $o(t)$ between them and their master clocks at time $t$, as follows:

$$
o(t) = s(t) - m(t)$$

where $s(t)$ is the time measured at the slave clock at time $t$, and $m(t)$ is the time measured at the master clock at time $t$. And the offset time is the time correction offered by the PTP protocol. The $s(t)$ and $m(t)$ are computed by sending PTP packets among the networking devices.

Though PTP is one of the time synchronization protocols, which is used in 4G and 5G technologies, for higher precision, it has some shortcomings. For example in PTPv2.1, the threat of compromised transparent clocks, delay attacks, etc [25]. Moreover, the symmetric time error assumption in PTP protocol does not hold true because of the presence of differences in the length of cables, etc [3].
2.3 Mobile network as a graph

From the above description of a mobile network with PTP protocol as a time synchronization technique, we can visualize it in the form of a graph, where the network consists of Grandmaster, routers, and switches, which consist of boundary clocks, and base stations as the nodes of the graph, where the grandmaster nodes are the root and base stations are the leaves of the graph. An example to describe it is shown in Fig. 2.2, where 'gm' represents the grandmasters, 'bc' represents the boundary clocks, and 'bs' represents the base stations. Details about how time error difference is calculated in this network are described in detail in Chapter 3.

Another term that would be used while discussing the results in Chapter 4 is 'number of steps' or 'number of hops', which is the number of networking devices present between the grandmaster and the base station.

2.4 Reinforcement learning

Reinforcement Learning (RL) has emerged as a transformative paradigm in the field of artificial intelligence, enabling autonomous agents to learn and make decisions in dynamic and uncertain environments. A Markov Decision Process (MDP) is a mathematical framework used to model and formalize the problem statement in reinforcement learning. In other words, RL is a way of solving MDP when the dynamics of the system are
unknown. MDP consists of the following components:

- **States ($S$):** A finite or infinite set of possible conditions the agent can find itself, and each element $s$ where, $s \in S$, is the representation of the environment received by the agent [26]. The state contains all the information that makes the decision-making independent of history.

- **Actions ($A$):** A set of actions the agent can take in each state. In other words, action $a$, where $a \in A_s$, is the action the agent would take when in state $s$ [26].

- **Transitions ($P$):** A probabilistic model that defines the probability distribution of transitioning from one state to another based on the action taken. In mathematical terms, it’s represented as $P(s'|s, a)$, which gives the probability of transitioning to state $s'$ from state $s$ when action $a$ is taken.

- **Rewards ($R$):** A function that assigns a numerical reward to each state-action pair. In mathematical terms, it’s represented as $R(s, a, s')$, indicating the reward obtained when transitioning from state $s$ to state $s'$ by taking action $a$.

- **Discount Factor ($\gamma$):** A value between 0 and 1 that represents how much the agent values future rewards compared to immediate rewards. It accounts for the trade-off between short-term and long-term gains.

- **Policy ($\pi$):** A policy defines the agent’s strategy or behavior, specifying which action to take in each state. That is the probability of the agent choosing a action $a$ given the state $s$, as shown in equation 2.1

$$\pi(a|s) = P(A = a|S = s) \quad (2.1)$$

The goal of an MDP is to find an optimal policy (a strategy) that maximizes the expected cumulative reward, $G$, equation 2.2. This is typically done using algorithms like dynamic programming [27], Monte Carlo methods [28], or temporal difference learning [29].

$$G = E \left[ \sum_{i=0}^{\infty} \gamma^i (R_i) \right] \quad (2.2)$$

RL leverages the principles of MDPs described above to find solutions to decision-making tasks. The agent is the decision maker in the RL process,
which observes the states, takes actions, and gets rewards or penalties based on its actions. The value function estimates the expected cumulative reward that an agent can obtain from a particular state or state-action pair. It helps the agent evaluate the goodness of different states or actions and guide its decision-making. There are two types of value functions: state-value function $V^\pi(s)$ and state-action value function $Q^\pi(s, a)$. The state-value function $V^\pi(s)$ estimates the expected cumulative reward starting from a specific state under a given policy. The action-value function $Q^\pi(s, a)$ estimates the expected cumulative reward starting from a state, taking a particular action a, and then following policy.

One possible solution to detect the fault is to train an RL agent to predict the time error difference values, and the links that cross the permissible time error range could be labeled as faulty. We chose to use Reinforcement Learning (RL) because we need to predict time error values for different links. To do this, we have to try out different time error values and create a way for the RL agent to understand the differences in time errors between two base stations. RL is a good choice for this because it allows us to explore and figure out the best strategies in situations that change over time. This makes RL a great fit for dealing with the complicated and uncertain task of predicting time errors. This is illustrated for better understanding in Fig. 2.3, where, the first figure towards the left represents known time error differences at the base stations represented with green and red links, and the rest all links with unknown asymmetries. By deploying an RL agent, it can predict the asymmetry values at each of the links, and those values when crossing a certain threshold, say 40 for the Fig. 2.3, those links would be considered as the spots with asymmetry.
2.5 Graph neural network

Graph Neural Networks (GNN) are one of the types of deep learning models, designed to deal with data represented in the form of graphs. Unlike the traditional dataset, where the data is organized in rows, graph data, unstructured and non-Euclidean, consists of a network of nodes and links connected to each other based on some relation, like social networks, mobile networks, etc. Traditional deep neural architectures fail to deal with this kind of complex data structure which lacks order. To deal with this GNNs were introduced, which use the Message passing algorithm and represent the graph in a structured data type to learn the graph, more about which is discussed in the section below.

The key components of the graph, the nodes and links could have different features. If the set of features is the same for all the nodes and the links, such graphs are called Homogenous graphs. If there are different kinds of nodes or links in the graph with different sets of features, then such a graph is called a heterogeneous graph. The tasks performed using GNN are node classification, link prediction, graph generation, and graph classification.

The goal of node classification is to predict the label or class of a particular node in the graph, for example predicting the labeling of the user based on their interest and behavior. The link prediction goal is to predict if there is a link between two given nodes, for example finding if two people know each other or not in a social network, based on their friends. In graph generation goal is to generate a new graph that shares similar properties with the input graph, for example, generating molecules with desired properties during drug discovery. In graph classification, the whole graph is associated with a label, for example, to predict the properties of chemical compounds based on their molecular structure. In this thesis work, we focus on node classification tasks. To perform these tasks, different frameworks of GNNs are used, which include GCN, GraphSAGE, HGNN, etc. The following sections are a detailed description of each of the different graph neural network types that were used in this thesis work.

Since the mobile network can be represented in the form of a graph, with each of the components having a defined set of features, GNN would be useful in making the model learn the graph structure, make predictions in the time error values, and predict the faulty node. The way to represent the mobile network as the homogenous and heterogeneous graph is described in Fig.2.5 and Fig.2.4 respectively, where the grey nodes represent the grandmaster, orange the boundary clocks, and blue nodes the base stations.
Figure 2.4: Sample of the Heterogenous graph generated for the customer dataset

Figure 2.5: Sample of the Homogeneous graph generated for the customer dataset
2.5.1 Working of GNN

Before understanding the workings of different GNN frameworks, we need to understand the general workings of the GNN.

Consider a graph $G(N, E)$, where $N$ is the set of all the nodes and $E$ links, with $n$ number of nodes and $e$ number of links. Also, each of the nodes has $f_1$ different features, and each of the links has $f_2$. This graph data is represented in the form of an adjacency matrix, and feature matrix.

An adjacency matrix is a square matrix (of dimension $n \times n$) used to represent the connection between nodes in the graph, where each row and column represents a node, and the entry at the intersection represents if there is a link between the corresponding two nodes. Value 1 in the cell $a_{ij}$ represents the presence of a link between node $i$ and $j$ and 0 means there is no link between them.

A feature matrix is of dimension $n \times f_1$. For link features, the adjacency matrix is appended with a matrix of the link feature for each link instead of a number, which results in the adjacency matrix having dimension $n \times n \times f_2$.

These matrices are passed to a neural network, where the information about each node is passed to the adjacent nodes, and this process is repeated iteratively, which results in all the nodes being aware of all other nodes present in the graph. In simpler terms, we can say that the working of GNN is analogous to CNN (Convolutional Neural Network) [30], where GNN is used for graphs and CNN for images.

Assume that the feature of each node is represented as $h_i$ for node $i$, this is the initial message vector that is passed from each node to the adjacent nodes. After collecting all the messages from the adjacent nodes, this information is aggregated using a message passing function $\psi$, and the feature vector $h_i$ is updated.

2.5.2 GCN

In Graph Convolutional Neural Network (GCN), during the message-passing algorithm, a node gets a message, which is a feature vector of all the adjacent nodes and aggregates it to get the update for its message, and sends the message to all the adjacent nodes. The mathematical representation of the message-passing algorithm is described below.

For each layer of the neural network $l$, for every node $i$, the message passed
$h_i^{(l)}$ is calculated as follows,

$$ h_i^{(l)} = \sigma \left( \sum_{u \in N(i)} \frac{h_i^{(l-1)}}{|N(i)|} \right) $$

where $N(i)$ represents the set of the neighbor nodes for node $i$, $w^{(l)}$, represents the weights in the neural network for layer $l$. $\sigma$ represents the activation function.

### 2.5.3 GraphSAGE

GraphSAGE focuses on inductive representation learning of graphs. This is similar to the GCN, except that each node receives and sends a message to a randomly selected set of adjacent nodes. This is advantageous in case we have huge graph data, as it results in a reduction in the number of computations, in comparison to GCN, hence leverages the node attribute information to efficiently generate representations on previously unseen data.

The mathematical representation of the message $h_i^{(l)}$, for node $i$, at layer $l$, is described below,

$$ h_i^{(l)} = \sigma \left( w^{(l)} \cdot \text{CONCAT} \left( h_i^{(l-1)}, \text{AGG}(\{h_u^{(l-1)}, \forall u \in N(i)\}) \right) \right) $$

where $N(i)$ represents the set of the neighbor nodes for node $i$, $w^{(l)}$, represents the weights in the neural network for layer $l$. $\sigma$ represents the activation function, AGG is a set of different aggregation functions one could use based on the type of task.

### 2.5.4 HGNN

Heterogenous Graph Neural Network (HGNN), is used to deal with heterogeneous graphs. This framework could be built with a combination of different kinds of GNN frameworks.

### 2.6 F1 score

The scoring function used to evaluate the models was the F1 score, with imbalanced datasets, and, for the classification task. The mathematical
formula for F1 score is

\[
F1 \text{ score} = \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]

where, recall and precision are computed as follows:

\[
\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}
\]

\[
\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}
\]

where, TP, represents True Positive, which represents the number of times the modal predicts the positive class, FP represents False Positive, which represents the number of times the modal incorrectly identifies the positive class and FN represents False Negatives, which represents the number of times the modal incorrectly identifies the positive class.

2.7 Dealing with imbalanced dataset

A dataset is considered imbalanced when there is a disproportionate number of data points for each label or class. Training model with imbalanced datasets results in poor performance in the minority class. To deal with it, various techniques are used. They are oversampling, undersampling, SMOTE, etc. Some of these techniques were used in this thesis work as the dataset provided was imbalanced. They are discussed in detail below.

2.7.1 SMOTE

The Synthetic Minority Oversampling TEchnique (SMOTE) is used to generate synthetic new data points for the minority class. It does this by selecting a random data point from the minority class and then generating new synthetic points along the line connecting that data point and its k nearest neighbors within the same minority class. These k nearest neighbors are found by calculating the Euclidean distance between them and selecting the ones with less value. Mathematically, the synthetic data points are created by taking the feature vector of the randomly chosen data point, subtracting it from one of its k nearest neighbors, and then scaling the result by a random number between 0 and 1. This technique makes the region of the minority class more general, hence helping the classifier to identify the minority class with better accuracy.
2.7.2 SMOTE-N

SMOTE-N (SMOTE-Nominal) is the SMOTE algorithm extended for nominal variables. The only difference in this method is the way the k neighbors are calculated. In this algorithm, the k nearest neighbors are selected based on 'Value Difference Metrics', which looks at the overlap of feature values overall features vectors. Mathematically this can be represented as $\delta$, is below:

$$
\delta(V_1, V_2) = \sum_{i=1}^{n} \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right|^k
$$

where $V_1$ and $V_2$ are the two corresponding feature values. $C_1$ is the total number of occurrences of feature value $V_1$, and $C_{1i}$ is the number of occurrences of feature value $V_1$ for class $i$. A similar convention can also be applied to $C_{2i}$ and $C_2$. $k$ is a constant, usually set to 1.

2.7.3 SMOTE-NC

SMOTE-NC (SMOTE-Nominal Categorical) is the SMOTE algorithm extended for nominal and categorical variables. This algorithm also differs in the computation of the distance between the neighbors. The way we compute the nearest neighbors is, first by computing the median of the standard deviation of all the continuous variables, If the nominal features differ between a sample and its potential nearest neighbors, then this median is included in the Euclidean distance. This median is used to penalize the difference in continuous feature values. Then compute the Euclidean distance in the feature space, and the rest of the steps remain the same as for SMOTE for the synthetic data point generation.

These techniques were used to balance the number of faulty links and non-faulty links present in the customer dataset.

2.8 Related works

Detecting of the faults to ensure time synchronization, has been of interest and there have been some studies that have shown significant improvement in time synchronization.

The source for faults in time synchronization algorithms, could not only be due to asymmetrical time error but also, from deliberate attacks. The Time
Synchronization Attacks (TSA) in Phasor Measurement Units (PMU), are one of the major threats to the modern smart grid applications [31], where, by compromising the time reference of a set of PMUs, the attacker can change the phase angle of the measured phasors. In research paper [31], TSAs against the PMU are detected using Graph Signal Processing (GSP) to model the power grid, to facilitate the detection and localization of TSA, using Random forests and Graph Fourier Transform. The research paper [32] proposes the use of a three-phase model for detecting TSA. The research paper [33] proposes a data-driven and model-based approach to detect TSA, where a phasor measurement model was developed to derive an accurate closed-form expression for the correlation between the frequency adjustments made by the PMU clock and the resulting change in the measured phase angle, without an attack. Then a model-based and three data-driven TSA detectors were employed to exploit the change in correlation due to a TSA.

In [34], AI-based synchronization scheme is proposed to achieve time synchronization to deal with the time synchronization issues in smart grid enabled technologies. The proposed AIFS scheme imposes a learning-based backpropagation technique for the PMUs in smart grid applications [34]. By extending the Flooding time synchronization protocol, to support fault tolerance in case of faulty nodes that lie about their global time, [35], proposes a novel algorithm, called Fault-Tolerant Flooding Time Synchronization Protocol for Wireless Sensor Networks (fault-tolerant FTSP). By detecting the time received before and the time received at the moment, the fault-tolerant FTSP detects an inconsistency and starts a decision process. The decision process has three steps: fault detection, asking for help, and receiving help and decision. After these three steps, the node determines if the timing information received is correct and can be stored or discarded [35].

### 2.8.1 GNN for classification and regression

GNNs perform three different types of tasks, node classification, link prediction, and graph classification. GNN has played an important role in drug discovery [36], understanding protein interactions [37], recommendations in social networks [38] etc. GNNs were used to perform node classification tasks, where the goal of the GNN is to predict if the node is faulty or not. Some of the previous work where GNNs were used to perform node classification in the telecommunication industry include, [39], where Graph Convolutional Gated Recurrent Neural Network was introduced to predict network-wide traffic. In addition to detecting the potential faulty elements, regression was applied
using GNNs to predict the asymmetry value, described further in Section 3.7. For building the regression model, RegGNN was explored. RegGNN is a graph neural network architecture for many-to-one regression tasks with application to functional brain connectomes (a comprehensive map for neural connection in the brain) for IQ score prediction [40]. The methods to predict the IQ value from brain connectomes, before RegGNN, relied on flattening the brain connectomes via vectorization and hence ignoring the topological properties. This method was considered similar to the current problem statement, as we need to predict the asymmetry value when we give the whole graph structure as input, in other words, modeling the asymmetry prediction, as a many-to-many problem.

In order to predict the asymmetry values, a GNN was trained to perform the classification task, and the node features were extracted and applied to the dataset to perform regression, which was inspiration from this paper.
Chapter 3

Methods

This chapter provides a step-by-step narrative of the methods used and the research conducted throughout the thesis. It elucidates the reasons and motivations behind conducting this study. This chapter delves into the methodologies utilized, the different experiments performed, and the rationale supporting the research choices made throughout the study.

The primary objective is to detect faulty components in the mobile network to ensure time synchronization. The faulty components are all those components that are responsible for contributing to the time error difference at the RAN domain.

The narrative of the major milestones accomplished during the thesis work is as follows:

- **Initial Hypothesis**: The thesis began by proposing the hypothesis that Reinforcement Learning (RL) could be employed to address the problem statement at hand.

- **Simulator Development**: A rule-based simulator was built, which was based on the rules suggested by domain experts. An intentional fault was introduced in this synthetic dataset in order to get a labeled dataset.

- **Failure of the hypothesis**: The various components for building an RL agent were assessed, and it was concluded that using RL to solve this problem statement is not the optimum solution.

- **Mathematical techniques for Fault detection**: Mathematical techniques as discussed in Chapter 2 were employed and the dataset was modeled in a unique way to solve the problem statement.
• **Labeling the unlabeled customer dataset**: For dealing with the unlabeled customer dataset, various classification labeling techniques were employed to assign relevant labels to the data, to enable better training and evaluation of GNN models.

• **GNN to get closer to the fault detection**: Further various GNNs were built to improve the faulty nodes detection, and various experiments were conducted as discussed in the sections below.

The following sections consist of a detailed description of each of the milestones described above. Before that, Section 3.1 provides the data structure of the mobile network.

### 3.1 Mobile network data structure

The components present in the mobile network are discussed in an elaborate way in Chapter 2. Here, we will dive deep into how the time error difference value for each cell pair is computed.

As mentioned earlier, the mobile network consists of the grandmaster as the root node and the base station as the leaf mode. The time error for each base station is the summation of all the time errors from the root to itself. These time errors add up across all the network components in between, like boundary clocks, routers, switches, etc. including the grandmaster itself.

The customer dataset consists of the time error differences, between two base stations, and those time differences are obtained by the OAS solution. However, it is also important to note that, the individual contribution of time error value from each of these networking components is unknown in the provided customer dataset. Also, we don’t have the individual time error value at each of the base stations. The time error difference between the two base stations is the only data available in the customer dataset, that is, for base station A and base station B, with time error say, $t_1$ and $t_2$ respectively, we know, $t_1 - t_2$, and also the time error difference between the base station B and A that is $t_2 - t_1$, as shown in Fig. 3.1.
3.2 Initial hypothesis

As mentioned in the previous section, the goal is to detect faulty network components in the mobile network. The mobile network can be represented in the form of a graph as shown in Fig. 2.5 and Fig. 2.4. So deploying an RL agent that can predict the intermediate time error values, which in other words are the unknown time error values as mentioned in the Section 3.1. If those time error values crossed a certain threshold, which makes the component faulty, then those devices could be labeled faulty. This is shown in Fig. 2.3. But to train an RL agent requires a lot of data, which was not available. In order to deal with this issue, a simulator was developed described below.

3.3 Simulator development

The rule-based simulator was developed based on the rules suggested by the domain experts. Along with it, an interactive web page was created for this and further studies. The reference for the web page, with the results, is mentioned below in Fig. 3.2

The rules that were used to build the simulator are listed below:
Figure 3.2: The screenshot of the web page interface for the simulator dataset.

- **Graph Layers**: The generated graph data consists of three sets of layers: a grandmaster layer, Boundary clock layers, and a base station layer.

- **Multi-layered Boundary clocks**: There could be more than 1 layer of Boundary clocks.

- **Connections among Boundary Clocks**: The simulator allows for connections between Boundary clocks, contributing to the network’s structural complexity.

- **Time Error Values**: The time error values for the grandmaster could be of range $(0 - 100)\text{ns}$, for boundary clocks the acceptable time error value range is $(5 - 30)\text{ns}$, for the base station, it is $(15 - 20)\text{ns}$, and each of the link it is $(15 - 20)\text{ns}$.

- **Introducing Error**: The error value of 1000ns is introduced to the fault percentage of network components, which is provided as input from the simulator.

- **More about connections**: Connections do not exist between the grandmaster and the base station.

The inputs that need to be provided to the simulator are:

- The number of grandmasters, boundary clocks, and base stations.
Figure 3.3: Simulator dataset versus the Customer dataset.

- The percentage of the links of all the possible combinations of links between the base stations or the number of the base station to base station links.
- Number of layers for boundary clocks
- Percentage of fault that needs to be introduced

The output that the simulator generates includes:

- A jpg image of the mobile network for reference.
- A 3d image of the graph on the web page, which is referenced from [41]
- A dataset, with all the information about the base stations, with columns
- A dataset, with two base stations and the time error differences between them, which resembles the customer dataset.

The advantages of the dataset generated from the simulator in comparison to the customer dataset are that we have the whole graph structure available, as shown in Fig.3.3 and we can introduce fault and classify the nodes as faulty and non-faulty as per the error introduced.

3.4 Failure of the hypothesis

Training an RL agent with its state space to all the possible time error values that could be present across each link in the graph, and action space to be
30 | Methods

Figure 3.4: Issues using RL

A range of possible values we can assign to each of the links present in the mobile network, could be a way to use RL to find the values of asymmetries in the mobile network. Here the reward function could be designed in such a way that if the asymmetries are defined when the time error differences at the base stations match with the actual time error difference between the base stations given to us, then we can award a positive reward.

However, the range of the acceptable time error values for each of the types of nodes, grandmaster, boundary clock, and base station is known in case of no error, however, if there is an erroneous link we don’t know the range of possible values for the node. Moreover, the graph structure varying for each of the mobile networks poses a challenge to the agent to adapt its policy for each new graph encountered during testing and training. Also, the agent can choose a time error value from a range, including decimals, which means a larger action space for the agent. All these contribute to making this approach more computationally expensive. Moreover, because there can be multiple valid combinations of values that yield the same time error difference, this doesn’t lead to the accurate determination of the asymmetry values. These reasons are further illustrated in the Fig. 3.4

It is also worth mentioning that, the network structure in itself is dynamic, which means, after a certain time, the graph structure of the mobile network changes, due to the selection of a new path as per the algorithm Best Master Clock Algorithm, which makes choosing RL as an option to solve this problem more challenging, as waiting for the agent to learn, would not help towards solving the problem.
3.5 Mathematical techniques for fault detection

After the failure of the hypothesis, various other techniques were explored. The mobile network which consists of base stations, boundary clocks and grandmasters is a graph with links between grandmasters and the boundary clocks, among boundary clocks and boundary clocks to base stations and among base stations. Here we know the time error values between the base stations, and the base stations in the customer dataset have node features as number of steps, feature 1 and feature 2 (confidential features at Ericsson), while the node features for the boundary clocks and the grandmaster is number of steps, as shown in Fig.2.4 and Fig.2.5. Now with each of the networking components as a variable with an unknown time error value, we can convert the graph to a set of equations.

The time error difference values for the base stations are the summation of all the time error values and then the difference, as mentioned in Section 3.1. So, it is possible to convert the time error difference values to a set of linear equations, where each of the network components from the root to the base station are variables, as shown in Fig.3.5.

Once we get those sets of equations, Gauss-Jordan elimination was employed to reduce the set of equations to a simplified version. This process reduced the number of equations to a significant level. These simplified sets of reduced equations were then analyzed to see which network components were potentially faulty.

In the reduced set of equations, the equations that had time error difference values more than zero, were filtered, and the occurrence of each of the networking elements was checked. If the networking elements occurred more than once, they were considered potentially faulty networking components. This method is summarized in the Fig.3.5.

However, it was found that the networking components which were labeled potentially faulty were a superset of the actually faulty elements, most of the time.

To check this, a simulator dataset with 16 graphs was generated with an error in it, and out of those 16 graphs, this observation matched 14 graphs. Further results about it are discussed in Chapter 4. So, it was important to check if other features of the networking devices could be used to get closer to the faulty components. Hence, a customer dataset was used, but the challenge with it was that it was an unlabelled dataset. So, the next task was to label that...
unlabeled dataset using the techniques mentioned in the following section.

3.6 Labeling customer dataset

The customer dataset consisted of time error difference values in the base station layer, along with information about the grandmaster parent port clock identity, the number of steps removed, and two additional confidential features for each of the base stations. The time error difference values were classified as non-faulty if they belonged to the [-1500, 1500]ns range, else faulty. However, the classification of each of the base stations was not performed. To classify the base stations or the other nodes, it was important to come up with the classification algorithms described below.

The algorithms are as described below:

- **Classification Algorithm 1**: Assume that there are $k$ time error differences from a particular node to all the nodes it is connected to. In that case, if more than 50% of the $k$ links are high time error values or faulty time error values, then the node is considered faulty. That is, consider out of these $k$ links if $m$ were faulty then,

  $$ s = \frac{m}{k} $$

  if score $s$, is greater than or equal to 0.5 then the node is faulty else non-faulty. This is illustrated in Fig. 3.6
Classification Algorithm 2: Another method was to rate the link based on its connection to the common connection of the two nodes, that is if the two nodes had a common boundary clock then the score was computed differently. For all the faulty links in the base station layer, the node was considered faulty, by assigning a score, based on if neighbor base stations have the same boundary clock then the score is incremented by 0.25, else if they both belong to the same graph then 0.5 and if to the different grandmasters subgraph 1.0 was assigned. If the error was greater than 0.0 then the node was considered faulty. That is the error coming from further away could indicate fault more, this is because, if all the nodes are faulty, then identifying time error is difficult. This is further illustrated with an example in Fig. 3.7

It’s worth emphasizing that these algorithms have a specific scope, which is the classification of nodes within the base station layer. Additionally, there was a proposal for a classification algorithm that would cover all nodes, but it wasn’t utilized in the study. The reason for not focusing on classifying other node types is that by successfully identifying faulty base stations and addressing timing errors at those stations, we can effectively resolve the core issue at hand. However, here’s a description of the algorithm for classifying all nodes: In this method, the classification of base station nodes follows the principles outlined in Classification Algorithm 2 mentioned earlier. If more than 50% of the base stations connected to a specific boundary clock are found to be faulty, that node is marked as faulty; otherwise, it is considered non-faulty. This same process is repeated for grandmasters, which are classified as faulty if more than
50% of their associated boundary clocks are faulty. Following this process, if a boundary clock is identified as faulty, all base stations connected to it are classified as non-faulty. Similarly, if a grandmaster is classified as faulty, all child nodes associated with that grandmaster are considered non-faulty. It’s important to acknowledge that these algorithms may not necessarily represent the optimal solutions for the problem statement, and there’s room for exploring additional classification algorithms as part of future research.

3.7 GNN models

After performing analysis on the customer dataset, it was found that there were 56 graphs in the dataset, and the information about the intermediate devices was absent in this dataset, as shown in Fig.3.3. So, in other to perform the mathematical techniques on it, the graphs were generated with available information by ignoring the intermediate devices. Further, the information about the number of intermediate devices between the grandmaster and the base station was present in the number of steps (which is described in Chapter 2).

A different number of experiments were conducted to check which other features were important for identifying the faulty cells. The experiments are described below:

- **Feature importance**: Creating a homogenous graph, as shown in
Fig. 2.5, with nodes as the base stations and node features as the grandmaster, boundary clock, the number of steps, and two other confidential features, GNN models were trained to classify the nodes as faulty or non-faulty with the help of the classification algorithms. Once the model was trained, it was checked which features the model gave more importance to, the results are discussed further in the next chapter. However, we considered grandmaster and boundary clocks as features to arrive at this conclusion. The possibility of using them as nodes has not been explored yet. So the next study was performed.

- **Node classification**: Furthermore, classification of the base station was performed, by creating a heterogenous graph, as shown in Fig. 2.4, with 3 different kinds of nodes, and features for the base station as the 2 features, number of steps, for grandmaster and boundary clocks it was number of steps.

- **Regression using GNN**: The model which best performed on the classification of nodes was chosen, ‘GCN for feature importance’, architecture in the next chapter(GNN architecture section), and the node embeddings for each of those nodes was extracted. These node embeddings were augmented to the dataset, with all the links present, that is gm(grandmaster) to bc(boundary clock), bc to bc, bc to bs(base stations), and bs to bs, in order to predict the time error values. However, this did not provide desirable results, more details about which are discussed in the Results and Discussions chapter.

### 3.8 Software tools and frameworks

This thesis work has been done using Python as the primary programming language. Various libraries were used for different tasks. The simulator was developed using Flask [42], a web application framework. For data handling and manipulation, the Pandas [43] and NumPy [44] libraries were utilized. The visualization of results was facilitated through the adoption of Matplotlib [45], Plotly [46], and Networkx [47]. To construct the Graph Neural Networks (GNNs), the Pytorch_geometric [48] library was used.
Chapter 4

Results and Discussion

In this chapter, we discuss all the qualitative and quantitative results for the methods that were used. The qualitative results consist of a discussion about how accurate were the labeling techniques, and more about the regression technique used to predict the node’s time error value. In the quantitative results discussion, the various neural network architectures are mentioned, along with the results from mathematical analysis and various GNN architecture experiments.

4.1 Data Collection and Data Preparation

During this thesis, the datasets used to perform various studies were either generated from the simulator built as a part of the study, or, the customer dataset provided by Ericsson. The customer dataset provided by Ericsson consisted of 56 sub-graphs with each node in the graph as a base station. For each base station node, the information about the parent port clock identity, or the nearest boundary clock to the particular base station, the grandmaster for the base station, the number of hops from the grandmaster to the base station, and two features of the base station were provided. Information about the pairs of base stations that were connected to one another was also provided. This dataset was unlabelled, which means, we did not have the label for faulty and non-faulty base stations. However, information about faulty links between a pair of base stations, measured using OAS protocol was provided. The links with the absolute value of time error difference greater than 1500ns were considered
faulty links, else were non-faulty. The simulator dataset used for the studies is described in the respective sections below.

4.2 Qualitative results

4.2.1 How accurate were the labeling techniques used?

To check for the correctness of each of the classification algorithms mentioned in Section 3.6, the simulator was used.

4.2.1.1 Simulator dataset and heatmaps

To obtain the value for each of the cell in the heatmaps in Fig.4.1, three random graphs were generated using the simulator. For the heatmaps in Fig.4.1b and Fig.4.1d, the number of base stations was fixed to 40, number of grandmasters was 5, number of layers among the boundary clocks was 3. The fault percentage and percentage of links were 10% and 10% respectively for the first three random graphs. The accuracy of the classification algorithm was measured by checking the percentage of correctly predicted faulty nodes by the algorithm. The average of these percentages is what each cell represents in the heatmap. For computing the values for the subsequent cells, the fault percentage and number of links percentage were gradually increased till 100%. Similarly, for the heatmaps in Fig.4.1a and 4.1c, the fault percentage was fixed to 50%, the number of grandmasters was 5, the number of layers among the boundary clocks was 3. The number of base stations was gradually increased from 10 to 100, and the number of links was increased from 10% to 100%. Three random graphs were generated for each of the values, and the average percentage of correctly identified cells was computed and represented in each of the heatmap cells.

The results for classification algorithm 1 are in Fig.4.1a and Fig.4.1b, and for classification algorithm 2 are in Fig.4.1c and Fig.4.1d.

4.2.1.2 Conclusions from the study

Some conclusions we can draw from these classification algorithms, which describe the motive for choosing one algorithm over another for further studies
are mentioned below:

- As the percentage of links among the base stations increases, the probability that the predictions are correct increases in classification algorithm 1. Refer Fig.4.1a

- A lower percentage of fault and a high percentage of links is ideal for getting better classification using both algorithms, refer to Fig.4.1b and Fig.4.1d.

- In the customer dataset, the average number of links among the base stations was found to be 44.6%

- Classification Algorithm 1 classifies the nodes better than Algorithm 2, as the given number of tries, the percentage of times the results were accurate are more in the case of algorithm 1, as seen via the cells with a greater number of brighter yellow shade in Fig.4.1 for algorithm 1

Hence, for future studies, Classification Algorithm 1 was used to perform the study.
4.2.2 GNNs for regression

A neural network which is a sequential network of 4 layers of 100, 50, 25, and 1 node in each of the layers, for 1000 epochs was used for regression on the dataset described in the Section 3.7 under the Regression using GNN. The loss function was customized to accommodate a range of permitted time error values including the error time error range along with the mean square error, and it is described in the function below.

\[
\text{mse\_loss} = \text{nn.MSELoss}()\left(\text{output}, \text{target}\right)
\]
\[
\text{in\_upper\_bound} = \text{torch.abs}(\text{self.upper\_bound} - \text{output})
\]
\[
\text{in\_lower\_bound} = \text{torch.abs}(\text{output} - \text{self.lower\_bound})
\]
\[
\text{range\_loss} = \text{torch.mean}\left(-\{\text{torch.min}(\text{in\_upper\_bound} + \text{in\_lower\_bound}, \text{torch.tensor}(0.0))\}\right)
\]
\[
\text{total\_loss} = \text{mse\_loss} + \text{range\_loss}
\]

However, it didn’t generate the results as expected, as the above-mentioned loss was 18000 for the first epoch and it continued to increase to 32000 over 1000 epochs. The assumption that this method could have failed is due to the presence of few features to learn from to perform regression. This could hopefully improve if the loss function is further modified such that not only a range of time errors but also a customized range for faulty and non-faulty nodes should be taken into consideration, along with increasing the number of features relevant to this problem.

4.3 Quantitative results

4.3.1 Mathematical methods results

The results generated using the Mathematical Methods described in Chapter 3 can be classified into two categories. They are results from the simulator dataset and results from the customer dataset.

4.3.1.1 Simulator dataset

A simulator dataset was generated with 16 sub-graphs with an average of 40% fault in each of the subgraphs and the percentage of links among the base stations to be 45%. The number of base stations was randomly chosen between 10 and 20, the number of boundary clocks was randomly selected between 4 to 16 and the number of grandmasters was between 3 and 10.
4.3.1.2 Results for simulator dataset

In the simulator dataset, the percentage reduction in the number of components from the initial number of components in the equation lists is 98.52%. The percentage of these which were correctly identified for each graph on average is 95.3125%. However, it is important to note that, for 14 of the 16 graphs this percentage was 100% and for the other two it was 95% and 25%. It is also important to note that, the set of the nodes detected as faulty by this technique were superset of the actual faulty nodes in 14 of the 16 graphs.

4.3.1.3 Results for the customer dataset

This method was then applied on the customer dataset, and the average reduction in the number of nodes for the customer dataset for each individual graph is 68.85%.

4.3.2 GNN architectures

Some of the GNN architectures, we used for this study are described below.

- **GCN for feature importance** Six layers GCN network made out of 20 hidden layers and relu activation [49], and output dimension with 2 layers and softmax as the activation for the final layer.

- **HGNN-Basic** [49] Two layers of HGNN network made out of GCN neural network for each type of link in the dataset, which is, gm to bc, bc to gm, bc to bs, and bs to bs, where the hidden dimension for both the layers is 16 with relu activation [49]. The optimizer is the Adam optimizer [50] and cross-entropy loss is to perform classification with the last layer as the linear layer with 2 nodes.

- **HGNN-using-GraphSage** Three Graph Sage layers with hidden dimension 16 and relu activation [49], for each of the types of links mentioned above. And then the last layer to perform classification with 2 nodes.

4.3.3 GNN experiments and results

In the ‘Feature Importance’ study mentioned in GCN model described in Section 4.3.2, was trained for 20 epochs on the customer dataset, the accuracy
of the model for the training dataset was 89.9%, and the feature importance graph is as shown in Fig. 4.2. From here it was clear that the two new customer features had no impact on the GNN’s learning. The grandmasters, boundary clock, and the number of steps removed had a huge impact on fault detection. By analyzing the feature importance of each of the base station’s features, it was found that no other feature other than the number of steps removed was found to be relevant for performing this study.

HGNN models, HGNN-Basic and HGNN-GraphSAGE were trained for 20 epochs on the customer dataset, and the models along with their best accuracy obtained by trial and error basis are mentioned in the table below, for the customer dataset HGNN-Basic with train and test accuracy as 95% and 93% respectively as shown in Table 4.1 for the customer data, with classification algorithm 1 with 20 epochs. However the trial and error-based search for the optimum number of nodes was performed by gradually increasing the number of hidden layers from 2 to 6 layers, and the number of hidden nodes from 5 to 30. Due to the presence of less complexity in the presented customer dataset, and the presence of lower features to train from, the model performed better for a lower number of iterations and with a smaller neural network architecture. Though the model performed equally better with the neural network architecture with a greater number of nodes in the hidden layer and a greater number of hidden layers, it was not preferred, in order to prevent overfitting of the data.
Table 4.1: F1 score (mentioned in Section 2.6) for the model in order to check for feature importance

<table>
<thead>
<tr>
<th>GNN Model</th>
<th>train F1 Score</th>
<th>test F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>HGNN-Basic</td>
<td>0.95</td>
<td>0.934</td>
</tr>
<tr>
<td>HGNN-using-GraphSage</td>
<td>0.945</td>
<td>0.934</td>
</tr>
</tbody>
</table>

4.4 Discussion about the research questions

This section is a discussion about the answers to the research questions mentioned in the Chapter 1, based on the study performed during the thesis.

- **What are the various techniques one could use to identify the faulty nodes or in other words, the most suitable nodes where one could apply asymmetry compensation?**  - The mobile network constituting of various network components can be visualized in form of a graph, as mentioned in Section 2.3. And each graph can be further simplified as a set of equations. Simplifying these set of equations, using methods like Gauss Jordan Elimination, unlocks the ability to detect faulty subnetworks present in the mobile network, and estimate the faulty nodes. Along with this non-machine learning approach, the Graph Neural Networks, the neural networks, that take the graph topology into account for making predictions, could also be used, in the presence of a labeled dataset with time error differences obtained using the OAS solution.

- **How accurate are these methods?**  The above-mentioned two approaches were tested on simulator datasets, and a customer dataset provided by Ericsson. The mathematical model, could successfully detect a superset of all the faulty elements, and the accuracy of finding all the faulty nodes in the superset was 95.3125% for the simulator dataset. For the customer dataset, according to this method, we can say that the faulty nodes are present in 31.15% of nodes detected by the mathematical approach.

With three different GNNs used for the classification of the faulty nodes, on the customer dataset, the accuracy of the models was 94.5% on the train data and 93.4% on the test dataset.
44 | Results and Discussion
Chapter 5

Conclusions and Future work

5.1 Conclusion

In conclusion, this study explored the use of Reinforcement Learning, Gauss-Jordan elimination, and Graph Neural Networks to detect asymmetries during time synchronization. It involved the development of a simulator and aimed at emulating real-world datasets. While Reinforcement Learning (RL) was initially considered but ultimately deemed unsuitable for our purposes, alternative techniques were explored. Specifically, we employed Graph Neural Networks (GNNs) and Gauss-Jordan Elimination.

By transforming the graph into a set of equations and subsequently applying Gauss-Jordan elimination, we successfully identified a superset of faulty nodes, achieving an impressive 95% accuracy on our simulated dataset. GNNs were not limited to node classification but also proved valuable in conducting feature importance analysis. This analysis revealed that two of the company’s features were irrelevant for detecting faults.

Furthermore, in the context of the customer dataset, Heterogenous Graph Neural Network (HGNN) models were used in identifying faulty nodes, and 94% accuracy on the test dataset was obtained. We also attempted regression using GNNs to predict asymmetry values. However, due to the limited number of features, this approach did not yield satisfactory results, leaving room for future exploration and enhancement.

Overall this research highlights the efficiency of GNNs and Gauss-Jordan Elimination in fault detection and feature analysis within simulated datasets, while also highlighting the need for further investigation into regression techniques with more comprehensive feature sets.
5.2 Limitations

Limitations of this work are presented below:

- The simulator selects the path at random and does not follow an algorithm like the Best Master Clock Algorithm in order to choose the next clock.

- This problem statement assumes that results are produced for a particular snapshot of time, instead of over a period of time, when there is a possibility that the links in the graph change over time.

- The correction value for the fault detection is not completely clear due to the presence of fewer features to learn from, and the scope of the project could greatly improve in the presence of more features and labeled datasets.

5.3 Future work

Some of the future works to proceed further could be:

- Classification of nodes is only performed for base stations, designing classification algorithms to perform classification to the nodes in the gm and bc layers could help in solving the problem statement.

- The simplified equations derived from the mathematical method can serve as a reliable means to approximate time error values effectively. Incorporating this information, alongside node embeddings during classification and additional features for learning, has the potential to be valuable in estimating time error values with good accuracy.

- Though the simulator dataset resembles closely to the real-world dataset, various other features could be added to make it more realistic like the Best Master Clock Algorithm to choose the master clock.

- This work could be further extended to dynamic graphs, which change over time.
Appendix A

Mathematical methods for solving equations

There are various mathematical methods, one could use to solve a system of equations. They are Gaussian elimination, Gauss-Jordan elimination, Matrix Inversion [51], Cramer’s rule [52], and Matrix decomposition methods like LU decomposition [53], Cholesky decomposition [54], QR decomposition [55], etc. In this thesis work the mathematical method used was Gauss-Jordan elimination described below. To understand Gauss Jordan elimination, it is also important to understand about Row echelon form and reduced row echelon form.

A.1 Row echelon form and Reduced row echelon form

A matrix is said to be in row echelon form if it satisfies the following conditions:

- The leftmost non-zero number in each row of the matrix is one (also known as the pivot).
- Pivots in each row are towards the right of the pivots in the rows above it.
- All the rows containing only zeros are toward the bottom of the matrix.
An example of a Row Echelon form of matrix is

\[
\begin{bmatrix}
1 & 2 & 3 \\
0 & 1 & 4 \\
0 & 0 & 1
\end{bmatrix}
\]

A matrix is said to be in Reduced row echelon form if it satisfies the following conditions:

- It satisfies all the conditions of the row echelon form of a matrix
- Each pivot is the only non-zero entry in the entire column

An example of a Reduced row Echelon form of matrix is

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

### A.2 Gauss Jordan elimination

Gauss-Jordan elimination is an efficient and systematic algorithm used to solve a system of linear equations and to find the inverse of square matrices. In this algorithm, we convert the set of linear equations to an augmented matrix and transform the matrix to a reduced row echelon form, by applying elementary operations iteratively, which gives us the solution to the equations. The elementary operations include interchanging two rows of the matrix and multiplying all the elements of a row by a non-zero value.

In simpler terms, it’s like a step-by-step process to transform a bunch of equations into a neat and easy-to-understand format. Imagine we have a set of equations as follows:

\[
\begin{align*}
2x + 3y &= 5 \\
4x - y &= 8
\end{align*}
\]

And we want to find the values of x and y that satisfy both equations. The Gauss-Jordan elimination process can help you do that systematically.

**Step 1: Create an Augmented Matrix** Firstly we create an augmented matrix. This is writing down the equations in a table format, where the coefficients of x and y, as well as the constants on the right side, are neatly organized. In our example, it would look like this:
Step 2: Row Operations

Now, we perform a series of operations on this matrix to simplify it. The three primary row operations are:

• **Scaling**: Multiplying a row by a constant

• **Row Addition/Subtraction**: Add or subtract rows with one another.

• **Row Swapping**: Swapping the positions of rows to get a more favorable arrangement.

These operations don’t change the solution to the equations; they just make it easier to find the variable values.

**Step 3: Reach Reduced Row Echelon Form**: Achieve reduced row echelon form described in Section A.1.

**Step 4: Read Off Your Solutions** Once we’ve achieved the reduced row echelon form, we can directly read off the solutions for the equations. In our example, after applying Gauss-Jordan elimination, we would end up with something like this:

\[
\begin{bmatrix}
1 & 0 & | & 2 \\
0 & 1 & | & -1
\end{bmatrix}
\]

which means that the solution of the equations is \( x = 2 \) and \( y = -1 \).

In summary, Gauss-Jordan elimination is a systematic method to solve systems of linear equations. The advantages of this method are, that it doesn’t need back substitution, which is needed in Gaussian elimination, and it provides accurate results.

By converting the mobile network components to the system of equations, and solving those system of equations, we can estimate which networking components could potentially be faulty.
References


